COVER Book Cover

VS FORTRAN Version 2

Installation and Customization for MVS

Release 6

Document Number SC26-4340-05

Program Number
5668-805
5668-806
5688-087

File Number S370-34

NOTICES Notices

+--- **Note!** --+
¦ ¦
¦ Before using this information and the product it supports, be sure ¦
¦ to read the general information under "Notices" in topic FRONT_1. ¦
¦ ¦
+---+

EDITION Edition Notice

Sixth Edition (November 1993)

This edition replaces and makes obsolete the previous edition, SC26-4340-04.

This edition applies to VS FORTRAN Version 2 Release 6, Program Numbers 5668-805, 5688-087, and 5668-806, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" following "About This Book." Specific changes for this edition are indicated by a vertical bar to the left of the change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

A Reader's Comment Form is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Canada Ltd. Laboratory, Information Development, 2G/345/1150/TOR, 1150 Eglinton Avenue East, North York, Ontario, Canada, M3C 1H7. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

CONTENTS Table of Contents

FRONT_1 Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectible rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

Subtopics

FRONT_1.1 Trademarks and Service Marks

FRONT_1.1 Trademarks and Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

AIX
AIX/ESA
BookMaster
ESA/390
ES/3090
ES/9000
IBM
MVS/ESA
MVS/SP
MVS/XA
RETAIN
SAA
Systems Application Architecture
System/370
System/390
VM/ESA
VM/XA

FRONT_2 About This Book

This book is intended to help you install, customize, and service VS FORTRAN Version 2 Release 6 under MVS. It is contains a description of the procedures for installation and customization.

Subtopics

FRONT_2.1 How This Book Is Organized

This book is organized as follows:

Chapter 1, "Getting Acquainted with VS FORTRAN Version 2," describes VS FORTRAN Version 2 and discusses where to find more information about it.

Chapter 2, "Planning for Installation," specifies the required systems and hardware, virtual storage and DASD space you will need when you install VS FORTRAN Version 2.

Chapter 3, "Setting Up SMP/E," describes how to set up either new or existing SMP/E data sets.

Chapter 4, "Installing VS FORTRAN Version 2," describes how to receive VS FORTRAN Version 2, and provides the procedures and data you need to install VS FORTRAN Version 2.

Chapter 5, "Making Interactive Debug Available to the User," provides instructions to make IAD available to ISPF/PDF users, users of ISPF without PDF, and non-ISPF users.

Chapter 6, "Customizing VS FORTRAN Version 2," describes how to customize cataloged procedures, make math library subroutines available, make load module AFBVRENT available, rebuild the composite modules, specify link or load mode, and change option defaults.

Appendix A, "Customization Macros," describes the customization macros and how to invoke them.

Appendix B, "Composite Modules," provides information about the required and optional components of the composite modules.

Appendix C, "Servicing VS FORTRAN Version 2," discusses problem reporting, corrective service, and preventive service.

FRONT_2.2 How to Use This Book

To install VS FORTRAN Version 2, you must complete the following:

1. Chapter 2, "Planning for Installation" in topic 2.0

2. Chapter 3, "Setting Up SMP/E" in topic 3.0

3. Chapter 4, "Installing VS FORTRAN Version 2" in topic 4.0

Following successful installation, you can perform the following optional activities:

Make VS FORTRAN Version 2 interactive debug available, if you purchased the complete product. Refer to Chapter 5, "Making Interactive Debug Available to the User" in topic 5.0. Separate procedures are provided for using ISPF with PDF, ISPF without PDF, as well as for non-ISPF users.

Customize VS FORTRAN Version 2 to your environment. Refer to Chapter 6, "Customizing VS FORTRAN Version 2" in topic 6.0; Appendix A, "Customization Macros" in topic A.0; and Appendix B, "Composite Modules" in topic B.0.

Subtopics
FRONT_2.2.1 Syntax Notation

FRONT_2.2.1 Syntax Notation

The following items explain how to interpret the syntax used in this manual:

Uppercase letters and special characters (such as commas and parentheses) are to be coded exactly as shown, except where otherwise noted. You can, however, mix lowercase and uppercase letters; lowercase letters are equivalent to their uppercase counterparts, except in character constants.

Italicized, lowercase letters or words indicate variables, such as array names or data types, and are to be substituted

Underlined letters or words indicate IBM-supplied defaults

Note: IBM is a trademark of the International Business Machines Corporation.

Ellipses (...) indicate that the preceding optional items may appear one or more times in succession

Braces ({ }) group items from which you must choose one

Square brackets ([ ]) group optional items from which you may choose none, one, or more

OR signs (|) indicate you may choose only one of the items they separate

FRONT_2.3 Publications

Figure 1 lists the VS FORTRAN Version 2 publications and the tasks they support.

Figure 1. VS FORTRAN Version 2 Publication Library

Task	**VS FORTRAN Version 2 Publication**	**Order Number**
Evaluation and Planning	*General Information*	GC26-4219
	Licensed Program Specifications	GC26-4225
Installation and Customization	*Installation and Customization for CMS*	SC26-4339
	Installation and Customization for MVS	SC26-4340
Application Programming	*Language and Library Reference*	SC26-4221
	Programming Guide for CMS and MVS	SC26-4222
	Interactive Debug Guide and Reference	SC26-4223
	Reference Summary	SX26-3751
Library Reference	*Master Index and Glossary*	SC26-4603
Diagnosis	*Diagnosis Guide*	LY27-9516
Migration	*Migration from the Parallel FORTRAN PRPQ*	SC26-4686

Figure 2 lists publications supply information you might need while you are installing, customizing or servicing VS FORTRAN.

Figure 2. Related Publications

Title	**Order Number**
MVS/SP Supervisor Services and Macro Instructions	GC28-1114
MVS/XA Supervisor Services and Macro Instructions	GC28-1154
SMP/E Messages and Codes	GC28-1108
SMP/E Reference Manual	SC28-1107
SMP/E User's Guide	SC28-1302
IBM High Level Assembler/MVS & VM & VSE: Language Reference	SC26-4940
IBM High Level Assembler/MVS & VM & VSE: Programmer's Guide	SC26-4941

FRONT_2.4 Industry Standards

The VS FORTRAN Version 2 compiler and library are designed according to the specifications of the following industry standards, as understood and interpreted by IBM as of December 1990.

The following two standards are technically equivalent. In the publications, references to **FORTRAN 77** are references to these two standards:

American National Standard Programming Language FORTRAN, ANSI X3.9-1978 (also known as FORTRAN 77

International Organization for Standardization ISO 1539-1980 Programming Languages--FORTRA

The bit string manipulation functions are based on ANSI/ISA-S61.1.

The following two standards are technically equivalent. References to **FORTRAN 66** are references to these two standards:

American Standard FORTRAN, X3.9-196

International Organization for Standardization ISO R 1539-1972 Programming Languages--FORTRA

At both the FORTRAN 77 and the FORTRAN 66 levels, the VS FORTRAN Version 2 language also includes IBM extensions. References to **current FORTRAN** are references to the FORTRAN 77 standard, plus the IBM extensions valid with it. References to **old FORTRAN** are references to the FORTRAN 66 standard, plus the IBM extensions valid with it.

Subtopics
FRONT_2.4.1 Documentation of IBM Extensions

FRONT_2.4.1 Documentation of IBM Extensions

In addition to the statements available in FORTRAN 77, IBM provides "extensions" to the language. In the *VS FORTRAN Version 2 Language and Library Reference*, these extensions are printed in color.

FRONT_3 Summary of Changes

Subtopics

FRONT_3.1 Release 6, October 1993

Subtopics
FRONT_3.1.1 Major Changes to the Product

FRONT_3.1.1 Major Changes to the Product

Support for AIX*/370 is not included in VS FORTRAN Version 2 Release 6. Support for AIX/ESA* is found in the AIX VS FORTRAN/ESA product.

Note: AIX and AIX/ESA are trademarks of the International Business Machines Corporation.

Printable copies of certain of the VS FORTRAN Version 2 Release 6 publications are provided on the product tape

Support for additional language has been added, much of which address previous incompatibilities between XL FORTRAN and VS FORTRAN

- Additional constant and operator support:
  - Quote-delimited literal character constants
  - Maximum negative integer literal value
  - Typeless constants (binary, octal and hexadecimal)
  - Named constants in complex constants
  - .XOR. logical operator
  - <> graphical relational operator
- Storage classes:
  - STATIC and AUTOMATIC storage classification statements
  - IMPLICIT STATIC and IMPLICIT AUTOMATIC statement support
- Additional data type support:
  - LOGICAL*2 and LOGICAL*8
  - INTEGER*1 and INTEGER*8
  - UNSIGNED*1
  - BYTE
  - DOUBLE COMPLEX and DOUBLE COMPLEX FUNCTION type specifications
- XREF and MAP Processing has been improved to remove storage-ordering perturbations.
- Additional intrinsic function support:
  - INTEGER*8, INTEGER*2, and INTEGER*1 support for existing functions
  - XOR, LSHIFT, RSHIFT, ISHFTC, IBITS, and LOC have been added.
- HALT compiler option to reduce compilation times for unsuccessful compilations
- MVBITS and ARGSTR service routines
- Dynamic storage support, for allocating and deallocating storage at run-time
  - Integer POINTER data type
  - Including dynamically dimensioned arrays
  - ALLOCATED, DEALLOCATE, and NULLIFY statements
  - Intrinsic function ALLOCATED
  - Compile-time option DDIM
- Dynamically loaded module support (via DYNAMIC option)
- I/O features enhancements:
  - Support for dummy arguments in NAMELIST lists.

¦ - NML keyword for NAMELIST READ/WRITE statements

¦ - OPEN and INQUIRE statement support for the POSITION, PAD, and DELIM specifiers

¦ - B and O format control codes

¦ - Expansion of the Z format control code

¦ - $ format control code

¦ Optimization improvements

¦ - Optimization for pipelined instruction scheduling
¦ - Improved optimization for Inter-loop register assignments

¦ Message improvements

¦ - All messages written to SYSTERM file
¦ - Information messages no longer marked as errors

¦ Run-time options improvements

¦ - Abbreviations are now allowed.

¦ - Default I/O units are now user-specifiable with ERRUNIT, RDRUNIT, PRTUNIT, and PUNUNIT

¦ Vector support enhancements

¦ - The VECTOR option defaults can be set at installation.
¦ - VECTOR suboptions MODEL and SPRECOPT.
¦ - Vectorized SIGN, ANINT, DNINT, NINT and IDNINT intrinsic functions
¦ - Vector performance improvements

¦ Parallel support enhancements

¦ - LOCAL statement allowing arrays for Parallel Do / Parallel Sections

¦ - NAMELIST processing for parallel environment allows local variables

¦ - Support for user-generated parallel trace (including service routines PTWRIT and PTPARM and suboption TRACE)

¦ - Parallel execution controls (affinity control)

¦ - Load module support for routines invoked via SCHEDULE and PARALLEL CALL

FRONT_3.2 Release 5, September 1991

Subtopics
FRONT_3.2.1 Major Changes to the Product

FRONT_3.2.1 Major Changes to the Product

Support has been added in the compiler and library for the Advanced Interactive Executive/370 (AIX/370) operating system. In addition, programs designed to be run on AIX/370 may use the following enhancements to VS FORTRAN Version 2:

- The **fvs** command to invoke the VS FORTRAN Version 2 compiler.
- Use of tab characters in source programs.
- Communication between Fortran programs and C programs.
- Coding of indirectly recursive Fortran routines.

Support for parallel programs has been added for programs running under CMS and MVS

Note: The VS FORTRAN Version 1 standard math routines (VSF2MATH) are not supported for parallel processing. Interactive debug can be used only with non-parallel programs, and with serial portions of parallel programs when the run-time option PARALLEL(1) is specified.

These enhancements support:

- Automatically generating parallel code for DO loops using a compile-time option.
- Explicit coding of parallel loops, sections, and calls with parallel language extensions.
- Using lock and event services to control synchronization in parallel programs.
- Directing the compiler to generate parallel or serial code using enhanced directives.
- Determining the number of virtual processors available and specification of the number of virtual processors to use during run time.
- Using I/O within parallel programs.
- Calling subroutines within parallel loops and sections.
- Obtaining information for tuning your parallel program using a compiler report listing.

Support for extended common blocks has been added for programs running under MVS/ESA* or VM/ESA*

Note: MVS/ESA and VM/ESA are trademarks of the International Business Machines Corporation.

Compile-time and run-time enhancements and options have been added to support the three types of common blocks that now exist.

Virtual storage constraint relief has been added for compiling under MVS/XA*, MVS/ESA*, VM/XA*, or VM/ESA*. This support allows the compiler to reside above the 16MB line and to process in 31-bit addressing mode. Therefore, larger programs can now be compiled.

Note: MVS/ESA, MVS/XA, VM/ESA, and VM/XA are trademarks of the International Business Machines Corporation.

The default name for a main program has been changed from MAIN to MAIN#, which allows MAIN to be used as a user name for a common block or other global entity.

Array declarator expressions for object-time dimensions can now be of type Integer*2

FRONT_3.3 Release 4, August 1989

Subtopics

FRONT_3.3.1 Major Changes to the Product

FRONT_3.3.1 Major Changes to the Product

Enhancements to the vector and optimization features of VS FORTRA
Version 2

- Automatic vectorization of user programs is enhanced by improvements to the dependence analysis done by the compiler. Specifically, the following constructs are eligible for vectorization:

  - Loops containing simple READ, WRITE, and PRINT statements
  - Loops containing equivalenced arrays
  - Loops containing branches out of the loop
  - Loop bound appearing in a subscript expression
  - Loop nests that process a triangular matrix
  - Simple IF loops
  - Integer sum reduction.

- Additional advanced vector optimization.

- In programs optimized at either OPT(2) or OPT(3), arithmetic constants will be propagated globally for scalar variables.

- In programs optimized at either OPT(2) or OPT(3), informational messages are issued when local scalar variables may be referenced before they have been initialized.

- Improved performance when the CHAR, ICHAR, and LEN character intrinsic functions are used.

- Single and double precision complex divide routines can be vectorized.

Enhancements to input/output support in VS FORTRAN Version

- Improved data transfer rate for sequential DASD and tape input/output on MVS systems.

- Ability to perform data striping (parallel I/O) on sequential data sets.

- Ability to detect input conversion and record length errors.

Enhancements to the intercompilation analyzer (ICA

Enhancements to the language capabilities of VS FORTRAN Version

- Ability to use graphic relational operators as an alternative to FORTRAN 77 relational operators.

Enhancements to the pseudo-assembler listing provided by VS FORTRAN Version

Enhancements to the math routines

- Single, double, and extended precision MOD library routines are more precise.

- Single and double precision scalar complex divide routines are more precise and faster, and are vectorizable.

- The old complex divide and MOD routines are in the alternate math library.

FRONT_3.4 Release 3, March 1988

Subtopics

FRONT_3.4.1 Major Changes to the Product

FRONT_3.4.1 Major Changes to the Product

Enhancements to the vector feature of VS FORTRAN Version

- Automatic vectorization of user programs is improved by relaxing some restrictions on vectorizable source code. Specifically, VS FORTRAN Version 2 can now vectorize MAX and MIN intrinsic functions, COMPLEX compares, adjustably dimensioned arrays, and DO loops with unknown increments.

- Ability to specify certain vector directives globally within a source program.

- Addition of an option to generate the vector report in source order.

- Ability to collect tuning information for vector source programs.

  Ability to record compile-time statistics on vector length and stride and include these statistics in the vector report.

  Ability to record and display run-time statistics on vector length and stride. Two new commands, VECSTAT and LISTVEC, have been added to interactive debug to support this function.

  Enhancements to interactive debug to allow timing and sampling of DO loops.

  Inclusion of vector feature messages in the online HELP function of interactive debug.

- Vectorization is allowed at OPTIMIZE(2) and OPTIMIZE(3).

Enhancements to the language capabilities of VS FORTRAN Version

- Ability to specify the file or data-set name on the INCLUDE statement.

- Ability to write comments on the same line as the code to which they refer.

- Support for the DO WHILE programming construct.

- Support for the ENDDO statement as the terminal statement of a DO loop.

- Enhancements to the DO statement so that the label of the terminal statement is optional.

- Support for statements extending to 99 continuation lines or a maximum of 6600 characters.

- Implementation of IBM's Systems Application Architecture* (SAA*) FORTRAN definition; support for a flagger to indicate source language that does not conform to the language defined by SAA.

  Note: Systems Application Architecture and SAA are trademarks of the International Business Machines Corporation.

- Support for the use of double-byte characters as variable names and as character data in source programs and I/O, and for interactive debug input and output.

- Support for the use of a comma to indicate the end of data in a formatted input field, thus eliminating the need for the user to insert leading or trailing zeros or blanks.

Enhancements to the programming aids in VS FORTRAN Version

- Enhancements to the intercompilation analysis function to detect conflicting and undefined arguments.

- Support for the data-in-virtual facility of MVS/XA and MVS/ESA.

- Ability to allocate certain commonly used files and data sets dynamically.

- Enhancements to the multitasking facility to allow large amounts of data to be passed between parallel subroutines using a dynamic common block.

- Support for named file I/O in parallel subroutines using the multitasking facility.

- Ability to determine the amount of CPU time used by a program or a portion of a program by using the CPUTIME service subroutine.

- Ability to determine the Fortran unit numbers that are available by using the UNTANY and UNTNOFD service subroutines.

Enhancements to the full screen functions of interactive debu

FRONT_3.5 Release 2, June 1987

Subtopics
FRONT_3.5.1 Major Changes to the Product

FRONT_3.5.1 Major Changes to the Product

Support for 31-character symbolic names, which can include the underscore (_) character

The ability to detect incompatibilities between separately-compiled program units using an intercompilation analyzer. The ICA compile-time option invokes this analysis during compilation.

Addition of the NONE keyword for the IMPLICIT statement

Enhancement of SDUMP when specified for programs vectorized at LEVEL(2), so that ISNs of vectorized statements and DO-loops appear in the object listing.

The ability of run-time library error-handling routines to identify vectorized statements when a program interrupt occurs, and the ability under interactive debug to set breakpoints at vectorized statements.

The ability, using the INQUIRE statement, to report file existence information based on the presence of the file on the storage medium.

Addition of the OCSTATUS run-time option to control checking of file existence during the processing of OPEN statements, and to control whether files are deleted from their storage media.

Under MVS, addition of a data set and an optional DD statement to be used during processing for loading library modules and interactive debug.

Under VM, the option of creating during installation a single VSF2LINK TXTLIB for use in link mode in place of VSF2LINK and VSF2FORT.

The ability to sample CPU use within a program unit using interactive debug. The new commands LISTSAMP and ANNOTATE have been added to support this function.

The ability to automatically allocate data sets for viewing in the interactive debug source window

FRONT_3.6 Release 1.1, September 1986

Subtopics

FRONT_3.6.1 Major Changes to the Product

FRONT_3.6.1 Major Changes to the Product

Addition of vector directives, including compile-time option (DIRECTIVE) and installation-time option (IGNORE

Addition of NOIOINIT run-time optio

Addition of support for VM/XA System Facility Release 2.0 (5664-169) operating syste

1.0 Chapter 1. Getting Acquainted with VS FORTRAN Version 2

Subtopics

1.1 What Is VS FORTRAN Version 2

VS FORTRAN Version 2 is an application programming tool, comprised of the following components:

The compiler, which translates programs written in the VS FORTRAN Version 2 language and produces object modules for subsequent running with the support of the VS FORTRAN Version 2 library.

The library, which contains mathematical, character, bit, service, input/output, and error routines. The library is designed to support all the features of the VS FORTRAN Version 2 language.

Interactive debug, which allows the programmer to monitor the running of VS FORTRAN Version 2 programs and to examine and change data at run time for non-parallel programs and the serial portions of parallel programs.

It is available as three separate products as stated below:

VS FORTRAN Version 2 (5668-806), the complete licensed program containing the compiler, library, and interactive debug. It also contains sample programs that allow you to verify the installation procedures.

VS FORTRAN Version 2 (5688-087), the compiler and library licensed program. It also contains a sample program that allows you to verify the installation procedures.

VS FORTRAN Version 2 Library (5668-805), a licensed program containing only the library

VS FORTRAN Version 2 is distributed by IBM Software Manufacturing and Delivery (ISMD). It is available on either a standard-labeled 9-track tape or a 3480 tape cartridge.

1.2 Where to Find More Information

The program directory, which accompanies the VS FORTRAN Version 2 product tape, contains additional information regarding program and service level information and supplemental installation notes.

A description of the functions supported by VS FORTRAN Version 2 can be found in the program announcement materials. Refer to *VS FORTRAN Version 2 Licensed Program Specifications (LPS)* or contact your IBM marketing representative for this information.

Updates to the information and procedures in this book may be obtained from either your IBM Support Center or through the RETAIN* Preventive Service Planning (PSP) Facility, as discussed on page 2.0.

Note: RETAIN is a trademark of the International Business Machines Corporation.

2.0 Chapter 2. Planning for Installation

This chapter helps you plan for the installation of VS FORTRAN Version 2. It specifies the required systems, hardware, virtual storage and DASD space.

Before installing VS FORTRAN Version 2 Release 6, contact your IBM Support Center or check the RETAIN/370 PSP (Preventive Service Planning) Facility for updates to this manual. To obtain this information from the PSP Facility, specify the following:

Upgrade	Complete Product Subset	Compiler and Library Product Subset	Library-only Product Subset
VSFORTRAN260	**HFR2602**	**HFT2602**	**HFL2602**
	JFR2611	**JFT2611**	**JFL2611**
	JFR2620	**HFL2602**	
	HFT2602	**JFL2611**	
	JFT2611	**JFT2612**	
	HFL2602	**JFT2613**	
	JFL2611		
	JFT2612		
	JFT2613		

Subtopics

#2.1 Installation Checklist

#The following checklist has been provided as an outline of the key activities involved in the installation process. The separate tasks #are listed in the sequence in which they should be completed.

Review supplied Program Director

Request and Review maintenance requirements from your IBM Support Center. In particular the Preventive Service Planning # (PSP) Bulletin should be reviewed.

Obtain sample install jobs and procedures from the supplied medi

Set up the SMP/E data set

Prepare the installation jobs and procedures for use (modify as necessary

RECEIVE produc

APPLY produc

Ensure necessary corrective service is applie

Run IVP and verify successful installatio

ACCEPT produc

#The next list of activities is optional. You may perform any or all of the tasks listed.

Make Interactive Debug available to user

Customize cataloged procedure

Make alternative math library subroutines availabl

Make the load module AFBVRENT availabl

Rebuild Composite Module

Specify the Mode of Run-Time Library Module

Change option default

Install the compiler in the Link Pack Area (LPA

2.2 Meeting the Basic Requirements

Operating system, hardware, virtual storage and DASD requirements are discussed in the following section.

Subtopics

2.2.1 Operating System Requirements

VS FORTRAN Version 2 Release 6 supports vector, parallel and scalar features as shown in Figure 3 in topic 2.2.1.

Figure 3. Compilation and Execution Support for Vector and Parallel Features

Product	**Vector**	**Parallel**	**Scalar/ Serial**
MVS/SP* Version 1 (5740-XYN or 5740-XYS) all releases, with or without TSO/E (5665-285)	O	O	X
Note: MVS/SP is a trademark of the International Business Machines Corporation.			
MVS/XA:			
MVS/SP Version 2 (5665-291 o 5740-XC6) all releases, and MVS/XA DFP Version 2 (5665-XA2) Release 3 or later, with or without TSO/E (5665-285)	O	X	X
MVS/SP Version 2 (5665-291 o 5740-XC6) Release 1.3 Vector Facility Enhancement or Release 1.7 or later, and MVS/XA DFP Version 2 (5665-XA2) Release 3 or later, with or without TSO/E (5665-285) Release 3 or later	X	X	X
MVS/ESA: MVS/SP Version 3 (5685-001 or 5685-002), MVS/ESA Version 4 (5695-047 or 5695-048) and MVS/ESA DFP Version 3 (5665-XA3), with or without TSO/E (5665-285) Release 3 or later or TSO/E Version 2 (5685-025)	X	X	X

Legend:

O indicates that compilation is supported for the specified feature.
X indicates that compilation and execution are supported for the specified feature.

Under MVS, VSAM support is included with the operating system environment as specified above.

Installation of VS FORTRAN Version 2 Release 6 using IBM-supplied jobs or system modification program extended (SMP/E) panels requires System Modification Program Extended (5668-949) Release 6 or later.

Assembler H Version 2 Release 1.0 or IBM High Level Assembler/MVS & VM & VSE is required for customization of VS FORTRAN Version 2 in MVS/XA and MVS/ESA environments.

Data-in-virtual processing requires MVS/SP Version 2 Release 2 (MVS/XA), MVS/XA DFP Version 2 Release 3 or MVS/SP Version 3 (MVS/ESA), and MVS/ESA DFP Version 3.

Interactive debugging requires TSO/E on MVS. Interactive debugging in full-screen mode requires ISPF Version 2 for MVS

(5665-319) with or without ISPF/PDF Version 2 for MVS (5665-317). For enhanced full-screen functions, 5665-319 is required.

In addition to the requirements given above for interactive debugging in full-screen mode, the appropriate ISPF/PDF product must be selected to use the following capabilities:

PDF browse and edit facilities in split-screen mod

Automatic browse of the debug print file and log at session en

Start of debugging by means of interactive debugging foreground invocation panel

2.2.2 Hardware Requirements

Programs compiled with the vectorization option specified are designed to run on the vector facility. All VS FORTRAN Version 2 Release 6 programs can be compiled, and all scalar programs will run on any IBM System/370*, System/390*, 43xx, 30xx, or 9370 Processor supported by the operating systems listed under the "Software Requirements" section above.

Note: System/370, and System/390 are trademarks of the International Business Machines Corporation.

The processor must have sufficient real storage to meet the combined storage requirements of the host operating system and access methods used.

The compiling and running of parallel programs can be done on a uniprocessor. However, to realize the possible performance improvements and reductions in elapsed time, a multiprocessor is required.

Subtopics
2.2.2.1 Machine Requirements
2.2.2.2 Virtual Storage Requirements
2.2.2.3 DASD Storage Requirements
2.2.2.4 Extended Common Requirements

2.2.2.1 Machine Requirements

The compile-time machine requires *both* of the following:

Any processing unit supported by MV
I/O devices used by the compiler (these are normally disks)

The run-time machine requires *all* of the following:

Any processing unit supported by MV
I/O devices used by the object program at run tim
An appropriate vector processor, if required by the object program at run time

Using extended common requires an ES/3090* Model S, Model J/JH, ES/9000*, or other ESA/390*-capable processor

Note: ESA/390, ES/3090, and ES/9000 are trademarks of the International Business Machines Corporation.

2.2.2.2 Virtual Storage Requirements

The VS FORTRAN Version 2 Compiler requires approximately 3 megabytes of virtual storage to handle a typical Fortran source program of 100 statements. The compiler can be loaded above the 16-megabyte line into the extended link pack area (ELPA). For more information on the extended link pack area, see "Installing the Compiler in the Link Pack Area" in topic 6.7.

Storage requirements for the VS FORTRAN Version 2 library vary according to the customization features selected, the size of user programs, the number of parallel tasks created and the number of virtual processors specified.

VS FORTRAN Version 2 interactive debug requires approximately 400K bytes of virtual storage to begin running, in addition to the storage required for the application program being debugged. The main storage requirements for debugging a program with VS FORTRAN Version 2 interactive debug depend on the function of your data. Interactive debug also acquires additional storage when the program is running; the amount varies according to the nature of the program being debugged and the type and quantity of debugging commands issued.

Parallel programs can require significantly more virtual storage than serial programs. If the application runs in an XA environment, the additional storage is acquired from above the 16-megabyte line if possible.

The RENT compile-time option and the separation tool can be used to reduce the virtual storage requirements.

2.2.2.3 DASD Storage Requirements

DASD storage requirements are based on the file devices that you plan to use. Determining DASD storage is discussed in the next section, "Identifying the VS FORTRAN Version 2 Data Sets."

2.2.2.4 Extended Common Requirements

VS FORTRAN Version 2 programs that use extended common blocks allocate storage for these blocks from MVS/ESA data spaces. To obtain this storage, VS FORTRAN must be able to allocate the necessary data spaces. Under MVS/ESA, data space allocation is controlled during installation using the IEFUSI exit routine, from the MVS/ESA Systems Management Facility (SMF).

The following are provided to assist you in determining these values:

1. VS FORTRAN allocates data spaces only as needed to contain extended common blocks. When the ECPACK run-time option is specified, each data space is allocated with a size large enough to hold the first extended common block placed into this data space up to a maximum of 2 gigabytes. As additional extended common blocks are used by the program, existing data spaces are expanded in an attempt to place the new extended common block into one of them. This expansion occurs only if the new actual size of the data space is within the limit established during installation by the IEFUSI exit routine.

   When an extended common block does not fit into an existing data space, a new data space is allocated, provided the number and limit of data spaces does not exceed the values specified in the IEFUSI exit. When these limits are exceeded, the Fortran program is terminated.

2. When the NOECPACK run-time option is specified, VS FORTRAN places each extended common block in its own data space, and the limits set by the IEFUSI exit must be sufficient to allow these data spaces to be defined. Because the size of each data space is limited to the size of the single extended common block within the data space, the size of each data space can be reduced.

   If you need different IEFUSI exit values for different users, you can consider job classes, job step program names, or some other mechanisms for determining the values for different jobs or TSO users.

2.3 Identifying the VS FORTRAN Version 2 Data Sets

Installation of VS FORTRAN Version 2 requires SMP/E and SMPTLIB data sets, target libraries, and distribution libraries. Prior to installation, you must:

1. Determine the amount of DASD storage required by the data sets and libraries. Figure 4, Figure 5, and Figure 6 in topic 2.3.1 will help you.

2. Determine where the data sets will be placed. Many sites place SMP/E and SMPTLIB data sets, target libraries, and distribution libraries on three different volumes to enhance performance. The target libraries are the only VS FORTRAN Version 2 libraries used on a regular basis. SMP/E and SMPTLIB data sets and distribution libraries are used only to install, customize and service VS FORTRAN Version 2.

3. Determine the data-set name prefix for each data set. Many sites choose a private prefix, such as VSF2, instead of SYS1, to distinguish these data sets from those required to IPL (initial program load) and run the system. (VSF2 is the default prefix for IBM-supplied jobs.)

   It is strongly recommended that you use the last qualifier of the data sets as given in the target and distribution library sections shown in "Target Libraries" and "Distribution Libraries" in topic 2.3.4. For example, if you choose VSF2 as a prefix, the compiler load modules target library would have a data set name of VSF2.VSF2COMP.

Subtopics

2.3.1 Determining Space Requirements for VS FORTRAN Data Sets

The following figures describe the DASD space requirements for installing VS FORTRAN. These are summary figures required for planning purposes. More detailed information is provided with the packaging materials shipped with VS FORTRAN Version 2. Figure 4 describes the DASD space requirements for the complete product: the compiler, the library, and interactive debug. Figure 5 describes the space requirements for the compiler and library and Figure 6 describes space requirements for the library-only. The figures list recommended aggregate allocations for the data sets, in tracks. These allocations include extra space to accommodate future maintenance activity.

Figure 4. FORTRAN Data Sets: Complete Product

	Estimated DASD Space (Tracks)		
Library Name	**3330**	**3350**	**3380/3390**
SMP/E(2)	790	540	230
SMPTLIB	1300	930	400
Target	1765	1245	535
Distribution	1390	980	420

(2) The space requirements for consolidated software inventory (CSI) are not included in this estimate.

Figure 5. FORTRAN Data Sets: Compiler & Library

	Estimated DASD Space (Tracks)		
Library Name	**3330**	**3350**	**3380/3390**
SMP/E(2)	290	260	125
SMPTLIB	835	570	255
Target	1110	725	330
Distribution	885	590	265

(2) The space requirements for consolidated software inventory (CSI) are not included in this estimate.

Figure 6. FORTRAN Data Sets: Library-Only

	Estimated DASD Space (Tracks)		
Library Name	**3330**	**3350**	**3380/3390**
SMP/E(3)	225	155	65
SMPTLIB	400	280	115
Target	720	485	210

Distribution	400	275	115

(3) The space requirements for consolidated software inventory (CSI) are not included in this estimate.

Subtopics
2.3.1.1 SMP/E Data Sets
2.3.1.2 SMPTLIB Data Sets

2.3.1.1 SMP/E Data Sets

If you elect to use existing SMP/E data sets, the aggregate space allocation in the preceding tables is an estimate of the additional tracks required to install VS FORTRAN Version 2. The estimates must be added to those of other products being applied that have modules or macros in the same SMP/E data sets.

You can allocate new SMP/E data sets by using one of the following jobs provided on your distribution tape:

Product	**Job**
Complete product	AFFESMPA
Compiler and library	ILXESMPA
Library-only	AFBESMPA

2.3.1.2 SMPTLIB Data Sets

MVS dynamically allocates SMPTLIB data sets during the SMP/E RECEIVE process. The DSSPACE subentry in SMP/E, shown in Figure 11 in topic 3.3, specifies the size for the largest SMPTLIB data set. The SMPTLIB data sets are created during RECEIVE and used during APPLY and ACCEPT. They are usually deleted after ACCEPT.

As each file is received from tape onto disk, unused space within the receiving data set is immediately released. The estimated DASD space requirements shown in Figure 4, Figure 5, and Figure 6 in topic 2.3.1 can assist you in planning SMPTLIB DASD storage requirements.

2.3.2 Target Libraries

The target libraries required to install the components of VS FORTRAN Version 2 are described below. For the aggregate DASD space requirements, see Figure 4, Figure 5, and Figure 6 in topic 2.3.1.

VSF2COMP Compiler load modules.

VSF2FORT Run-time library routines needed for creation of a program that can be run.

VSF2LINK Interface routines used in link mode only. This library must be concatenated ahead of VSF2FORT for the creation of a program ready to be run in link mode.

VSF2LOAD Routines that may be loaded when the program is running or when interactive debug is used.

VSF2MATH Alternative math routines, from VS FORTRAN Version 1 Release 4.

SAMPLIB Sample Fortran source programs and release migration tool.

PROCLIB Cataloged procedures to run VS FORTRAN Version 2 jobs.

HELP Interactive debug TSO Help.

VSF2TLIB Interactive debug table library.

VSF2PLIB Interactive debug ISPF panel library.

VSF2MLIB Interactive debug ISPF message library.

VSF2CLIB Interactive debug TSO CLIST library.

¦ SILXSPRT PARTRACE, Trace File record formats and ABFTRAC, sample formatting program

¦ SILXREAD User "readme" file with service-updated product information

¦ SILXPS PostScript formatted product publications

¦ SILX3820 BookMaster (*) formatted product publications

¦ (*) BookMaster is a trademark of the International Business Machines Corporation.

¦ *2.3.3 Product Libraries*

¦ Four of the target libraries are optionally renamed after the install is complete to the following names. If you wish to keep the target ¦ library names, skip this rename step.

¦ SUPPORT PARTRACE, Trace File record formats and ABFTRAC, sample formatting program

¦ README User "readme" file with service-updated product information

¦ LISTPS PostScript formatted product publications

¦ LIST3820 BookMaster formatted product publications

2.3.4 Distribution Libraries

The distribution libraries required to install the components of VS FORTRAN Version 2 are described below. For the aggregated DASD space requirements, see Figure 4, Figure 5, and Figure 6 in topic 2.3.1.

ILXCCM Compiler load modules.

ILXCCS Compiler macros, cataloged procedures, sample program, and SMP/E installation procedures.

AFBLBM Run-time library load modules.

AFBLBS Run-time library macros and SMP/E installation procedures

AFFMOD Interactive debug load modules.

AFFSRC Interactive debug TSO Help, sample programs, and SMP/E installation procedures.

AFFPLIB Interactive debug ISPF panel library.

AFFMLIB Interactive debug ISPF message library.

AFFCLIB Interactive debug ISPF CLIST library.

¦ AILXSPRT PARTRACE, Trace File record formats and ABFTRAC, sample formatting program

¦ AILXREAD User "readme" file with service-updated product information

¦ AILXPS PostScript formatted product publications

¦ AILX3820 BookMaster formatted product publications

When you have completed the installation planning activities, you can proceed with Chapter 3, "Setting Up SMP/E" in topic 3.0.

3.0 Chapter 3. Setting Up SMP/E

Proceed only if you have completed Chapter 2, "Planning for Installation" in topic 2.0.

At this point, choose to either set up new SMP/E data sets or use existing SMP/E data sets. If you want to either maintain an earlier release of VS FORTRAN Version 2 or keep SMP/E isolated, establish new SMP/E data sets. Alternatively, if you want to minimize the number of SMP/E data sets on your system by grouping Program Products or languages together, use the existing SMP/E data sets.

Installation of VS FORTRAN Version 2 Release 6 will delete previous releases of Version 2. However, you can avoid this deletion by specifying different SMP/E, distribution, and target library data sets from those used for previous releases, or by renaming your existing SMP/E, distribution, and target library data sets. If you choose to specify different SMP/E, distribution, and target library data sets, you will need to make the same changes in the remaining steps of the installation process.

Subtopics

3.1 Obtaining Sample Installation Jobs and Procedures

Sample installation jobs have been provided on the distribution tape with your VS FORTRAN Version 2 product. These sample jobs #may be modified to suit your installation's requirements and run to install your VS FORTRAN Version 2 product. In particular, you will #need to increase the directory space by at least ten percent.

Subtopics
3.1.1 Running GETPROCS Job

3.1.1 Running GETPROCS Job

The six SMP/E sample installation jobs and procedure are shown in Figure 7.

Figure 7. SMP/E Sample Installation Jobs and Procedure

Job Name	Job Function	Procedure Invoked
aaaESMPA	Allocates SMP/E data sets	In line
aaaESMPI	Initializes SMP/E data sets	In line
aaaERECV	Receives VS FORTRAN Version 2	In line
aaaALOC	Allocates data sets needed to install VS FORTRAN Version 2	In line
aaaEAPPL	Applies VS FORTRAN Version 2	**aaa**EPROC
aaaEACPT	Accepts VS FORTRAN Version 2	**aaa**EPROC

In this table, **aaa** is **AFF** if you are installing the complete product, **ILX** if you are installing the compiler and library product, and **AFB** if you are installing the library-only product.

By running the job shown in Figure 8, the six SMP/E sample installation jobs and the SMP/E procedure used by two of them will be copied from the product tape to disk. The system will then catalog them in a data set and print a listing. The listing is useful if you must modify either the JCL or SMP/E statements.

Before you run the JCL, choose the set of SMP/E sample installation jobs that match the VS FORTRAN Version 2 product you are installing. Then set up the **SELECT MEMBER** statement accordingly by selecting the appropriate **aaa** prefix. Notes at the bottom of Figure 8 will help you make these and other required changes.

```
-------------------------------------------------------------------------

  //GETPROCS JOB  .... (user information)
  //*
  //*   GET SMP/E INSTALLATION PROCEDURES FROM ISMD TAPE
  //*
  //        EXEC PGM=IEBCOPY
  //SYSPRINT DD  SYSOUT=A
  //IN       DD  DSN=filename,UNIT=3400-6,VOL=SER=volser,     Note 1, 2
  //             DISP=SHR,LABEL=(nn,SL)
  //OUT      DD  DSN=VSF2.INSTALL,                           Note 3
  //             DISP=(NEW,PASS),SPACE=(TRK,(1,1,2)),
  //             UNIT=SYSDA,VOL=SER=volid,                   Note 4
  //             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
  //SYSUT3   DD  SPACE=(TRK,(1)),UNIT=SYSDA
  //SYSUT4   DD  SPACE=(TRK,(1)),UNIT=SYSDA
  //SYSIN    DD  *
   COPY INDD=IN,OUTDD=OUT
   SELECT MEMBER=(aaaESMPA,aaaESMPI,aaaERECV,aaaEPROC)        Note 5
   SELECT MEMBER=(aaaALOC,aaaEAPPL,aaaEACPT)
  //*
  //*   PRINT THE PROCEDURES
  //*
  //    EXEC PGM=IEBPTPCH
```

```
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSN=VSF2.INSTALL,                          Note 3
//             DISP=(OLD,CATLG)
//SYSUT2   DD  SYSOUT=A
//SYSIN    DD  *
      PRINT TYPORG=PO,MAXFLDS=1
      RECORD FIELD=(80)
/*
```

Figure 8. GETPROCS Job to Receive SMP/E Data Sets

Notes

1. Specify **filename**, **volser** and nn as shown in Figure 9.

Figure 9. 'filename', 'volser' and 'nn' Specification

Product	**filename**	**volser**	**nn**
Complete product	JFR2611.F1	FR2602	5
Compiler and library	JFT2611.F1	FT2602	5
Library-only	JFL2611.F2	FL2602	6

2. Check UNIT=3400-6 for appropriateness to your site.

3. The SMP/E sample installation jobs will be copied from the tape into the PDS VSF2.INSTALL. If you want the jobs to be copied into a different data set, you must appropriately change the OUT DD statement and the SYSUT1 DD statement.

4. **Volid** is the volume serial number of the DASD for the sample jobs data set.

5. Specify the **SELECT MEMBER aaa** prefix as shown in Figure 10.

Figure 10. SELECT MEMBER aaa Prefix

Product	**aaa**
Complete product	AFF
Compiler and library	ILX
Library-only	AFB

3.2 Setting Up New SMP/E Data Sets

Proceed with this section only if you are setting up new SMP/E data sets.

Subtopics

3.2.1 Editing and Running SMP/E Jobs

Edit the following common and individual items:

Common item

You can change these procedure keyword defaults in the SMP/E jobs. Be sure to make the changes consistently in the two jobs, **aaa**ESMPI and **aaa**ESMPA.

SMPPRFX = VSF2 Data set name prefix for all SMP/E data sets

SMPVOL = VSFRES Serial number of the volume for SMP/E data sets including SMPTLIB data sets

UNIT = SYSDA Type of device on which this volume is mounted

SOUT = * SYSOUT class to use for all printable output

Individual item

The job statements in each of these jobs may have to be changed to meet your site's requirements. In particular, you will need to
increase the directory space by at least ten percent.

AFFESMPA | ILXESMPA | AFBESMPA
If you change the SMPPRFX default, make the corresponding change in the cluster names in the SYSIN DD statements. If you change the SMPVOL default, make the corresponding change in the VOLUME parameter in the SYSIN statement.

AFFESMPI | ILXESMPI | AFBESMPI
If you change the SMPPRFX default, make the corresponding change in the cluster names in the ZONEINDEX parameter. If you change the SMPPRFX default, you may also want to consider making a corresponding change in the DSPREFIX parameter.

Note: If you are installing the complete product, change the jobs beginning with **AFF**. If you are installing the compiler and library product, change the jobs beginning with **ILX**. If you are installing the library-only product, change the jobs beginning with **AFB**.

Run the following:

1. The **aaa**ESMPA job. The return code should be zero, indicating success. If you have a return code other than zero, consult *SMP/E Messages and Codes*. Possible problems are the following:

   Inconsistent values for UNIT and VOLSER parameters
   Insufficient space on the volume requested
   Inconsistent changes to the data set prefixes

   Before rerunning the job, delete any data sets that were partially created.

2. The **aaa**ESMPI job. The return code should be zero, indicating success. If you have other return codes, consult *SMP/E Messages and Codes*. Possible problems are the following:

   Failure of previous job
   Inconsistent changes to the data set prefixes
   Changes in SMPCNTL statements that accidentally deleted periods or parentheses

   This job cannot be rerun without cleanup.

After successful running of the setup jobs for new SMP/E data sets, proceed with "Preparing the Installation Jobs and Procedures for Use" in topic 3.4.

3.3 Setting Up Existing SMP/E Data Sets

Proceed with this section only if you are setting up existing SMP/E data sets.

Sample installation jobs are provided for SMP/E only. If your previous installation was in SMP4 data sets, you must use the new SMP/E data sets. To install VS FORTRAN Version 2 using existing SMP/E data sets, update the SMP/E entries as shown in Figure 11.

Figure 11. SMP/E Entries to be Updated for Existing SMP/E Data Sets

SMP/E ENTRY	VALUE
SREL	Z038
DSSPACE	(200,20,120)
PEMAX	(9999)
LKED	PARM(SIZE=(1024K,32K),NCAL, LET, XREF, LIST)
ASM	NAME(IEV90)
SAVEMTS	No value specified

After setting up existing SMP/E data sets, proceed with "Preparing the Installation Jobs and Procedures for Use," which follows.

3.4 Preparing the Installation Jobs and Procedures for Use

1. If you use IBM-supplied jobs, search each job for the word "optional" and update the corresponding statement as instructed by the comments.

2. The procedures define variables by using symbolic parameters. If you use IBM-supplied jobs, the variable definitions you supply in the jobs define the symbolic parameters in the procedures they invoke. If you use your own jobs, you can allow the symbolic parameters to default to the definitions provided on the PROC statements of the procedures, or you can override the defaults by using the EXEC statements that invoke the procedures from your jobs.

3. Update the following keyword defaults in **aaa**EPROC and in **aaa**ALOC:

   SMPPRFX='VSF2' Data set name prefix used for all SMP/E data sets.

   DISPRFX='VSF2' Data set name prefix used for all distribution library data sets.

   TARPRFX='VSF2' Data set name prefix used for all target library data sets.

   SAMPRFX='VSF2' Data set name prefix used for the SAMPLIB target library data set. This data set must have been previously allocated. This is **not** applicable if you are installing the **library-only** product.

   PROPRFX='VSF2' Data set name prefix used for the system PROCLIB target library data set. This data set must have been previously allocated. It is **not** applicable if you are installing the **library-only** product.

   HELPPRFX='VSF2' Data set name prefix used for the system HELP target library data set. This data set must have been previously allocated. It is **not** applicable if you are installing the **compiler and library** product or the **library-only** product.

   UNIT='SYSDA' Type of device on which the SMPTLIB data set volume is mounted and which is to be used when allocating the SMP/E work data sets.

   SMPVOL='VSFRES' Volume serial number (volser) of the volume that contains, or is to contain, the SMPTLIB data set.

   DISVOL='VSFRES' Volser of volume that contains distribution libraries.

   TARVOL='VSFRES' Volser of volume that contains target libraries.

   SOUT='*' SYSOUT class to use for all printable output.

4. After updating **aaa**EPROC, either insert it into a PROCLIB, or copy it into the installation jobs that invoke it (**aaa**EAPPL and **aaa**EACPT). If you copy **aaa**EPROC into the jobs, you must place it after the JOB statement and before the EXEC statement that calls it, and add the following statement after the last line of the procedure:

```
// PEND
```

3.5 Running the aaaALOC Job

1. Search for the word OPTIONAL. If you wish to allocate the SAMPLIB, PROCLIB, and HELP data sets, change the commented line **STEP 2** to a line that will be active in the program by deleting the "*".

2. Run the appropriate **aaa**ALOC job as shown below. The return code should be zero, and you should check the messages in the output to be sure that the specified data sets have been cataloged.

Job	**Allocates Data Sets For**
AFFALOC	Complete product
ILXALOC	Compiler and library product
AFBALOC	Library-only product

Proceed with Chapter 4, "Installing VS FORTRAN Version 2" in topic 4.0.

4.0 Chapter 4. Installing VS FORTRAN Version 2

The installation process is the same for the complete product, the compiler and library product, and the library-only product; however, there is no IBM-supplied verification program available for the library-only product.

Installation of VS FORTRAN Version 2 Release 6 will delete previous releases of Version 2. However, you can avoid this deletion by specifying different SMP/E, distribution, and target library data sets from those used for previous releases, or by renaming your existing SMP/E, distribution, and target library data sets. If you choose to specify different SMP/E, distribution, and target library data sets, you will need to make the same changes in the remaining steps of the installation process.

Proceed only if you have completed Chapter 2, "Planning for Installation" in topic 2.0 and Chapter 3, "Setting Up SMP/E" in topic 3.0.

Subtopics

4.1 RECEIVING VS FORTRAN Version 2

You can RECEIVE VS FORTRAN Version 2 by running the aaaERECV job obtained by the GETPROCS job, or by using SMP/E panels shown on page 4.1. The aaaERECV job will let you RECEIVE the product you ordered.

Method 1: Running the aaaERECV job:

1. Change the procedure parameters as appropriate.

2. Check UNIT=parameter on the SMPPTFIN DD statement for appropriateness to your site.

3. Run the appropriate job as shown below.

Job	**Applies To**
AFFERECV	Complete product
ILXERECV	Compiler and library
AFBERECV	Library-only

After successful running, you can proceed with "APPLYING VS FORTRAN Version 2" in topic 4.2.

Method 2: Using SMP/E panels:

Specify the following highlighted items:

1. SMP/E Primary Option menu:

```
                ===>  4   (Command Generation)
      MASTER SMPCSI ===>  'VSF2.SMPE.CSI' (or the name of your CSI)
      Generate DD   ===>  YES
```

2. Command Generation - Selection menu:

```
                ===>  12  (Receive)
      ZONE NAME ===>  GLOBAL
```

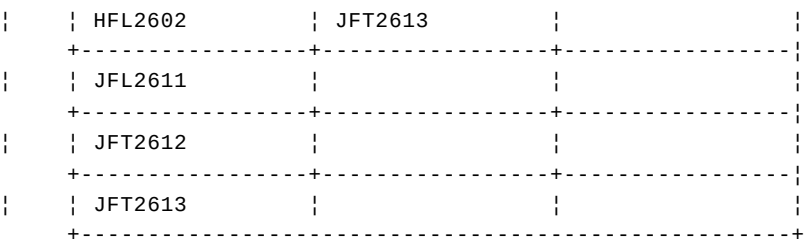
Note: If you are using SMP/E Release 5, Receive is option 10.

3. Command Generation - Receive Options menu:

```
      HOLDDATA  ===>  NO
      ALL       ===>  NO
      SELECT    ===>  YES
```

4. Command Generation - Receive Select List menu:

Complete Product	**Compiler and Library Product**	**Library-only Product**
HFR2602	HFT2602	HFL2602
JFR2611	JFT2611	JFL2611
JFR2620	HFL2602	
HFT2602	JFL2611	
JFT2611	JFT2612	



5. Command Generation - Receive Data Set Information menu:

```
DATA SET NAME ===> 'SMPMCS'
VOLUME SERIAL ===> FR2602        (See note)
UNIT          ===> 3400-6        (or as appropriate)
LABEL TYPE    ===> SL
FILE          ===> 1
```

¦ **Note:** For the compiler and library, use **FT2602**; or for the library-only use, **FL2602**.

6. Command Generation - Submit menu:

```
                ===>  S   (Submit)
```

   A 4-megabyte region is required for this step. Also, you must either have DDDEF entries for all SMP/E libraries, or add the appropriate DD statements to the job.

4.2 APPLYING VS FORTRAN Version 2

You can APPLY VS FORTRAN Version 2 by running job **aaa**EAPPL or by using SMP/E panels.

Method 1: Running the aaaEAPPL job:

1. The SMPCNTL APPLY statements that follow the procedure invocation show the FMIDs to be applied during installation. Refer to the job prologue or to Figure 42 in topic C.1 for a list of FMIDs.

2. Run the appropriate **aaa**EAPPL job shown in Figure 12.

Figure 12. aaaEAPPL Jobs and Procedure

Job	Procedure Invoked	Applies to
AFFEAPPL	AFFEPROC	Complete product
ILXEAPPL	ILXEPROC	Compiler and library
AFBEAPPL	AFBEPROC	Library-only

¦ You will receive message **IEW0461** from the linkage editor (or IEW2454 if using the binder) during the apply step of the library ¦ because of unresolved library routines. A condition code of 4 will result from SMP/E, and a condition code of 4 will result from the linkage editor. These messages and condition codes are normal and can be ignored.

After successful running, proceed with "Verifying Successful Installation" in topic 4.3.

Method 2: Using SMP/E panels:

Specify the following highlighted items:

1. SMP/E Primary Option menu:

```
                ===>  4   (Command Generation)
      MASTER SMPCSI ===>  'VSF2.SMPE.CSI' (or the name of your CSI)
      Generate DD   ===>  YES
```

2. Command Generation - Selection menu:

```
                ===>  11  (Apply)
      ZONE NAME     ===>  TVSF2  (or a name you choose)
```

3. Command Generation - Apply menu:

```
      FUNCTIONS     ===>  NO
      SELECT        ===>  YES
```

4. Command Generation - Apply Select List menu:

Complete Product	Compiler and Library Product	Library-only Product
HFR2602	HFT2602	HFL2602

JFR2611	**JFT2611**	**JFL2611**
JFR2620	**HFL2602**	
HFT2602	**JFL2611**	
JFT2611	**JFT2612**	
HFL2602	**JFT2613**	
JFL2611		
JFT2612		
JFT2613		

5. Command Generation - Submit menu:

```
                ===>  S   (Submit)
   Add  TIME=5  to your job statement.
```

A 4-megabyte region is suggested for this step. Also, you must have DDDEF entries for all SMP/E data sets, target libraries, and distribution libraries.

After successful running, proceed with "Verifying Successful Installation" in topic 4.3.

4.3 Verifying Successful Installation

If you are installing the library-only product, verification of installation is not a valid procedure. To verify successful installation of the complete product or the compiler and library product, run the compiler and library verification program AFBIVP provided on the
#product tape. Prior to running AFBIVP, you will need to apply the PTF **UN57464** and any other service which has been indicated.
#Once this has been completed successfully, you can proceed with the installation verification. Figure 13 is a sample JCL that runs the compile-link-go procedure, VSF2CLG; it will compile, link and run the verification program AFBIVP.

```
-------------------------------------------------------------------------

  //AFBIVP      JOB . . .
  //FORTRAN     EXEC VSF2CLG
  //FORT.SYSIN  DD DSN=VSF2.SAMPLIB(AFBIVP),DISP=SHR


-------------------------------------------------------------------------
```

Figure 13. Job to Verify Successful Installation of Product

During the APPLY step of installation, VSF2CLG is copied into VSF2.PROCLIB and AFBIVP is copied into VSF2.SAMPLIB. To run VSF2CLG, either:

Put it in a system PROCLIB, o

Copy it into the job after the JOB statement and before the EXEC statement. Remember to add ***// PEND*** after the last line of the procedure.

Note: The procedure VSF2CLG uses a prefix of SYS1 for the VS FORTRAN data sets. If you used the IBM-supplied jobs, these data sets will have a prefix of VSF2; therefore, you may need to edit the VSF2CLG procedure.

Successful running of the sample program causes the following message to be issued:

```
'SAMPLE PROGRAM COMPLETED SUCCESSFULLY'
```

4.4 ACCEPTING VS FORTRAN Version 2

You can ACCEPT VS FORTRAN Version 2 by running job **aaa**EACPT or by using SMP/E panels.

Method 1: Running the aaaEACPT job:

When you are satisfied that VS FORTRAN Version 2 is operating correctly, run this job to store the product in your system's distribution libraries (DLIBs). This job also deletes all SMPTLIB data sets associated with the installation of VS FORTRAN Version 2 Release 6.

1. The ACCEPT statements that follow the SMPCNTL DD statement define the FMIDs to be accepted. Refer to Figure 42 in topic C.1 for the list of FMIDs to be accepted. Make sure that the list in the **aaa**EACPT job matches the list of FMIDs you received and applied with the **aaa**ERECV and **aaa**EAPPL jobs.

2. Run the appropriate **aaa**EACPT job shown in Figure 14. If you are installing in distribution libraries that have not previously contained VS FORTRAN Version 2, you may receive message **GIM2471** for modules AFBVUAT and ILX0OPTS. These messages may be **ignored.**

Figure 14. aaaEACPT Jobs and Procedure

Job	**Procedure Invoked**	**Accepts**
AFFEACPT	AFFEPROC	Complete product
ILXEACPT	ILXEPROC	Compiler and library
AFBEACPT	AFBEPROC	Library-only

Installation of VS FORTRAN Version 2 is now complete. You may now proceed with Chapter 5, "Making Interactive Debug Available to the User" in topic 5.0 or Chapter 6, "Customizing VS FORTRAN Version 2" in topic 6.0.

Method 2: Using SMP/E panels:

Specify the following highlighted items:

1. SMP/E Primary Option menu:

```
                ===>  4   (Command Generation)
     MASTER SMPCSI ===>  'VSF2.SMPE.CSI' (or the name of your CSI)
     Generate DD   ===>  YES
```

2. "Command Generation - Selection" menu:

```
                ===>  12  (Accept)
     ZONE NAME     ===>  DVSF2  (or a name you choose)
```

3. Command Generation - Accept menu:

```
     FUNCTIONS     ===>  NO
     SELECT        ===>  YES
```

4. Command Generation - Accept Select List menu:

```
   +-----------------------------------------------------+
   ¦ Complete        ¦ Compiler and    ¦ Library-only    ¦
```

Product	Library Product	Product
HFR2602	HFT2602	HFL2602
JFR2611	JFT2611	JFL2611
JFR2620	HFL2602	
HFT2602	JFL2611	
JFT2611	JFT2612	
HFL2602	JFT2613	
JFL2611		
JFT2612		
JFT2613		

5. Command Generation - Submit menu:

```
            ===>  S   (Submit)
   Add  TIME=5  to your job statement.
```

A 4-megabyte region is suggested for this step. Also, you must have DDDEF entries for all SMP/E data sets, target libraries and distribution libraries.

Installation of VS FORTRAN Version 2 is now complete. You may now proceed with Chapter 5, "Making Interactive Debug Available to the User" in topic 5.0 or Chapter 6, "Customizing VS FORTRAN Version 2" in topic 6.0.

5.0 Chapter 5. Making Interactive Debug Available to the User

This is an optional step. If you wish to proceed, make sure you have completed all steps described under Chapter 4, "Installing VS FORTRAN Version 2" in topic 4.0.

If you have previously installed interactive debug for a prior release of VS FORTRAN Version 2, you may want to go directly to "Verifying Availability of Interactive Debug" (page 5.1.5 for users of ISPF/PDF, page 5.2.5 for users of ISPF without PDF, and page 5.3.2 for non-ISPF users), and proceed to "A Sample Interactive Debug Session" in topic 5.4.

Note: Interactive debug can be used only with non-parallel programs, and the serial portions of parallel programs when the run-time option PARALLEL(1) is specified.

Subtopics

5.1 Instructions for ISPF/PDF Users

Proceed only if you are using Interactive System Product Facility (ISPF) Version 2 and ISPF Product Development Facility (ISPF/PDF) Version 2. If you are using ISPF Version 2 without PDF, go to "Instructions for Users of ISPF Without PDF" in topic 5.2. If you are not using ISPF Version 2, go to "Instructions for Non-ISPF Users" in topic 5.3.

To use ISPF with PDF, complete the following required steps. Each step is described in more detail in the following sections, along with a procedure for verifying successful installation of interactive debug.

1. Modify the Foreground Selection Panel to provide the VS FORTRAN Version 2 interactive debug option, if the panel does not already include this option.

2. Modify the Help Panel to provide the VS FORTRAN Version 2 interactive debug option, if the panel does not already include this option.

3. Modify the Compilation CLIST to accommodate VS FORTRAN Version 2 programs.

4. Create an invocation procedure or CLIST to invoke ISPF/PDF.

Subtopics

5.1.1 1. Modifying the Foreground Selection Panel

Using ISPF/PDF Edit, modify foreground selection panel ISRFPA, located in your site's ISPF/PDF panel data set. Do **one** of the following steps:

Change:

1. Any option at the top part of the panel to specify VS FORTRAN Version 2 interactive debug.

2. The corresponding numbered line at the bottom part of the panel to specify AFFFP11.

For example, as shown in Figure 15, you can change option 11 from the old Fortran interactive debug product to VS FORTRAN V2 interactive debug, and change entry 11 at the bottom of the panel from ISRFP11 to AFFFP11 (enter the data in uppercase).

Add:

1. **%xx+- VS FORTRAN V2 Interactive Debug**

   to the list of options at the upper part of the panel (where xx is the number of the option).

2. **xx,'PGM(ISRFPR) PARM(AFFFP11) NEWAPPL(AFF)'**

   to the entries at the lower part of the panel. During installation, AFFFP11 will be included in the library containing the ISPF/PDF panel definitions of VS FORTRAN Version 2 interactive debug (VSF2PLIB).

```
+-------------------------------------------------------------------------------------+
¦                                                                                     ¦
¦                                                                                     ¦
¦                                                                                     ¦
¦                                                                                     ¦
¦   %-----------------------  FOREGROUND SELECTION PANEL  --------------------------   ¦
¦   %OPTION  ===>_ZCMD                                                          +     ¦
¦   %                                                                                 ¦
¦   %   1+- System assembler                % 7+- Linkage editor                     ¦
¦   %   2+- OS/VS COBOL compiler            % 8+- Load                               ¦
¦   %   3+- VS FORTRAN compiler             % 9+- SCRIPT/VS                          ¦
¦   %   4+- PL/I checkout compiler          %10+- COBOL Interactive Debug            ¦
¦   %   5+- PL/I optimizing compiler        %11+- VS FORTRAN V2 Interactive Debug    ¦
¦   %   6+- PASCAL/VS compiler              %12+- Member parts list                  ¦
¦   %                                                                                 ¦
¦   +                                                                                 ¦
¦   +SOURCE DATA PACKED%===>_ZFPACK  +(YES or NO)                                     ¦
¦   )INIT                                                                           ¦
¦     .HELP = ISR40000                                                              ¦
¦     IF (&ZXPACK ¬= '        ')                                                    ¦
¦       &ZFPACK = &ZXPACK                                                           ¦
¦       &ZXPACK = '        '                                                        ¦
¦     &ZFPACK = TRANS(TRUNC(&ZFPACK,1),Y,YES,*,NO) /* DATA FORMAT CHECK  */         ¦
¦   )REINIT                                                                         ¦
¦     &ZFPACK = TRANS(TRUNC(&ZFPACK,1),Y,YES,*,NO) /* DATA FORMAT CHECK  */         ¦
¦   )PROC                                                                           ¦
¦     &ZFPACK = TRUNC(&ZFPACK,1)                                                    ¦
¦     VER (&ZFPACK,NB,LIST,Y,N)          /* Y=EXPAND PACKED DATA        */         ¦
¦     &ZFPACK = TRANS(TRUNC(&ZFPACK,1),Y,YES,N,NO)                                  ¦
¦     VPUT (ZFPACK) PROFILE                                                         ¦
¦     &ZSEL = TRANS( TRUNC (&ZCMD,'.')                                              ¦
¦                 1,'PGM(ISRFPR) PARM(ISRFP01) NEWPOOL'                             ¦
```

```
                    2,'PGM(ISRFPR) PARM(ISRFP02) NEWPOOL'
                    3,'PGM(ISRFPR) PARM(ISRFP03) NEWPOOL'
                    4,'PGM(ISRFPR) PARM(ISRFP04) NEWPOOL'
                    5,'PGM(ISRFPR) PARM(ISRFP05) NEWPOOL'
                    6,'PGM(ISRFPR) PARM(ISRFP06) NEWPOOL'
                    7,'PGM(ISRFPR) PARM(ISRFP07) NEWPOOL'
                    8,'PGM(ISRFPR) PARM(ISRFP08) NEWPOOL'
                    9,'PGM(ISRFPR) PARM(ISRFP09) NEWPOOL'
                   10,'PGM(ISRFPR) PARM(ISRFP10) NEWPOOL'
                   11,'PGM(ISRFPR) PARM(AFFFP11) NEWAPPL(AFF)'
                   12,'PGM(ISRFPR) PARM(ISRFP12) NEWPOOL'
                  ' ',' '
                    *,'?' )
   )END
```

Figure 15. ISPF/PDF Foreground Selection Panel

5.1.2 2. Modifying the Help Panel

Using ISPF/PDF Edit, follow the procedures in "1. Modifying the Foreground Selection Panel" in topic 5.1.1 to either change or add to foreground help panel ISR40000 (located in your site's ISPF/PDF panel library). The corresponding numbered line at the bottom of the panel should specify AFF41100. See Figure 16.

```
%TUTORIAL -------------- FOREGROUND PROCESSING OPTION ----------------- TUTORIAL
%OPTION  ===>_ZCMD                                                          +
+
%                     -----------------------------------
                      |     FOREGROUND PROCESSING       |
                      -----------------------------------
+
   The foreground processing option allows certain processing programs to
   be run in the foreground under ISPF.  The foreground selection panel
   which is displayed when option%4+ is entered on the primary option menu
   allows the selection of one of these processing programs.

 The following topics are presented in sequence, or may be selected by number:
    %0+- Foreground general information   % 6+- PASCAL/VS compiler
    %1+- System assembler                % 7+- Linkage editor
    %2+- OS/VS COBOL compiler             % 9+- SCRIPT/VS
    %3+- VS FORTRAN compiler              %10+- COBOL Interactive Debug
    %4+- PL/I checkout compiler           %11+- VS FORTRAN V2 Interactive Debug
    %5+- PL/I optimizing compiler         %12+- Member parts listing

)PROC
   &ZSEL = TRANS( &ZCMD
                 0,ISR40001
                 1,ISR41000
                 2,ISR42000
                 3,ISR43000
                 4,ISR44000
                 5,ISR45000
                 6,ISR46000
                 7,ISR47000
                 9,ISR49000
                10,ISR4A000
                11,AFF41100
                12,ISR4C000
                 )
   &ZUP = ISR00003
)END
```

Figure 16. ISPF/PDF Foreground Processing Help Panel

5.1.3 3. Modifying the Compilation CLIST

In order to compile VS FORTRAN Version 2 programs through the ISPF/PDF environment, modify the CLIST ISRFC03, which is found in your location's ISPF/PDF CLIST library, as follows:

Replace the line

```
  ISPEXEC SELECT PGM(FORTVS)
```

with

```
  ISPEXEC SELECT PGM(FORTVS2)
```

5.1.4 4. Creating an Invocation Procedure or CLIST

1. Create a procedure or CLIST to invoke ISPF/PDF, by **one** of the following:

   Build or modify the TSO logon procedure. See Figure 17 for a sample logon procedure. If VSF2COMP and VSF2LOAD are in LNKLST, then it is not necessary to include them in STEPLIB.

   Use an editor to build or modify a CLIST. This CLIST must include ALLOC statements to allocate the libraries created during installation of VS FORTRAN Version 2 interactive debug. (See Figure 18 for an example of a CLIST containing the required names.)

   CAUTION: We advise you to use the procedures in this manual to establish your interactive debug libraries; they are the only ones supported by IAD. If you use another approach, such as the ISPF LIBDEF service, you may get unpredictable results.

2. The data sets VSF2CLIB, VSF2MLIB, VSF2PLIB, and VSF2TLIB are distributed in fixed block format. If this format is incompatible with the format of the concatenated data sets, you may have to modify the attributes of the data sets so proper concatenation will occur.

   For example, to modify the attributes of the VSF2.VSF2CLIB data set, take the following steps:

   a. Allocate a new data set for VSF2CLIB with the correct attributes and a different name.

   b. Copy the contents of VSF2CLIB into your new data set.

   c. Delete the original copy of VSF2CLIB.

   d. Rename the new data set VSF2CLIB.

```
//IADUSER EXEC PGM=IKJEFT01,DYNAMNBR=64,REGION=1024K
//*
//* IF VS FORTRAN VERSION 2'S VSF2.VSF2LOAD WAS INSTALLED IN A LIBRARY
//* WHICH IS INCLUDED IN LINKLIST (MEMBER LNKLSTxx OF SYS1.PARMLIB),
//* THE STEPLIB STATEMENT FOR VSF2.VSF2LOAD MAY BE DELETED.
//*
//STEPLIB   DD DSNAME=VSF2.VSF2COMP,DISP=SHR
//          DD DSNAME=VSF2.VSF2LOAD,DISP=SHR
//          DD DSNAME=ISP.V2R2M0.ISPLLIB,DISP=SHR
//          DD DSNAME=ISR.V2R2M0.ISRLLIB,DISP=SHR
//SYSEDIT   DD DSNAME=&&EDIT,UNIT=SYSDA,SPACE=(1688,(50,20))
//SYSEDIT2  DD DSNAME=&&EDIT2,UNIT=SYSDA,SPACE=(4096,(20,10))
//SYSIN     DD TERM=TS,SYSOUT=A
//SYSPRINT  DD TERM=TS,SYSOUT=A
//SYSTERM   DD TERM=TS,SYSOUT=A
//FT05F001  DD TERM=TS
//FT06F002  DD TERM=TS
//*
//* IF THE HELP MEMBER FOR VS FORTRAN VERSION 2 INTERACTIVE DEBUG
//* WAS NOT INSTALLED IN SYS1.HELP, CONCATENATE THE CORRECT DATA SET
//* NAME TO THE SYSHELP STATEMENT BELOW.
//*
//SYSHELP   DD DSNAME=SYS1.HELP,DISP=SHR
//SYSLBC    DD DSNAME=SYS1.BRODCAST,DISP=SHR
//FT05F001  DD TERM=TS,SYSOUT=A
```

```
//FT06F001  DD TERM=TS,SYSOUT=A
//*
//* YOU MAY HAVE TO MODIFY THE FOLLOWING LIBRARY NAMES FOR YOUR INSTALLATION.
//*
//ISPPLIB   DD DSNAME=VSF2.VSF2PLIB,DISP=SHR
//          DD DSNAME=ISP.V2R2M0.ISPPLIB,DISP=SHR
//          DD DSNAME=ISR.V2R2M0.ISRPLIB,DISP=SHR
//ISPMLIB   DD DSNAME=VSF2.VSF2MLIB,DISP=SHR
//          DD DSNAME=ISP.V2R2M0.ISPMLIB,DISP=SHR
//          DD DSNAME=ISR.V2R2M0.ISRMLIB,DISP=SHR
//ISPSLIB   DD DSNAME=ISP.V2R2M0.ISPSLIB,DISP=SHR
//          DD DSNAME=ISR.V2R2M0.ISRSLIB,DISP=SHR
//ISPTLIB   DD DSNAME=VSF2.VSF2TLIB,DISP=SHR
//          DD DSNAME=ISP.V2R2M0.ISPTLIB,DISP=SHR
//          DD DSNAME=ISR.V2R2M0.ISRTLIB,DISP=SHR
//SYSPROC   DD DSNAME=ISP.V2R2M0.ISPCLIB,DISP=SHR
//          DD DSNAME=ISR.V2R2M0.ISRCLIB,DISP=SHR
//          DD DSNAME=VSF2.VSF2CLIB,DISP=SHR
```

Figure 17. TSO LOGON Procedure

```
PROC 0
  CONTROL NOLIST NOSYMLIST NOCONLIST  NOFLUSH  NOMSG
   /*****************************************************************/
   /*                                                               */
   /* THIS IS AN EXAMPLE OF A CLIST FOR SETTING UP ISPF/PDF, WHICH  */
   /* INCLUDES DEFINITIONS FOR THE DATA SETS REQUIRED BY VS FORTRAN */
   /* VERSION 2 INTERACTIVE DEBUG.  THE ISPF/PDF NAMES SHOWN IN THIS */
   /* EXAMPLE MAY BE DIFFERENT FROM THE ONES USED AT YOUR LOCATION. */
   /*                                                               */
   /*****************************************************************/
   FREE  FI(SYSPROC ISPMLIB ISPPLIB ISPLLIB ISPTLIB)
   ALLOC FI(SYSPROC) DA('VSF2.VSF2CLIB' +
                    'ISP.V2R2M0.ISPCLIB' +
                    'ISR.V2R2M0.ISRCLIB' ) SHR REUSE
   ALLOC FI(ISPMLIB) DA('VSF2.VSF2MLIB' +
                    'ISP.V2R2M0.ISPMLIB' +
                    'ISR.V2R2M0.ISRMLIB' ) SHR REUSE
   ALLOC FI(ISPPLIB) DA('VSF2.VSF2PLIB' +
                    'ISP.V2R2M0.ISPPLIB' +
                    'ISR.V2R2M0.ISRPLIB' ) SHR REUSE

   ALLOC FI(ISPTLIB) DA('VSF2.VSF2TLIB' +
                    'ISP.V2R2M0.ISPTLIB' +
                    'ISR.V2R2M0.ISRTLIB' ) SHR REUSE

   /*****************************************************************/
   /*                                                               */
   /* THERE IS NO SKELETON FILE FOR VS FORTRAN VERSION 2 INTERACTIVE */
   /* DEBUG.  THE NORMAL ALLOCATION FOR ISPSLIB CAN BE USED.        */
   /*                                                               */
   /* IF THE HELP MEMBER FOR VS FORTRAN VERSION 2 INTERACTIVE DEBUG  */
   /* WAS NOT INSTALLED IN SYS1.HELP, THE CORRECT DATA SET NAME     */
   /* SHOULD BE CONCATENATED TO SYS1.HELP.  FOR EXAMPLE:            */
   /* FREE FI(SYSHELP)                                              */
```

```
/* ALLOC FI(SYSHELP) DA('SYS1.HELP' +                              */
/*                   'XXXX.HELP') SHR REUSE                        */
/*                                                                 */
/*****************************************************************/
ALLOC FI(ISPLLIB) DA('VSF2.VSF2COMP' +
                  'VSF2.VSF2LOAD' +
                  'ISP.V2R2M0.ISPLLIB' +
                  'ISR.V2R2M0.ISRLLIB' ) SHR REUSE
/*****************************************************************/
/*                                                               */
/* ALLOCATE SYSTEM INPUT AND OUTPUT FILES TO THE TERMINAL.       */
/* THIS CAN BE CHANGED BY USING TSO COMMANDS ANY TIME BEFORE     */
/* INVOKING THE PROGRAM.                                         */
/*                                                               */
/*****************************************************************/
ALLOC FI(FT05F001) DA(*)
ALLOC FI(FT06F001) DA(*)
```

Figure 18. ISPF/PDF CLIST

5.1.5 5. Verifying Availability of Interactive Debug

Use the interactive debug verification program AFFIVP provided on the product tape. During the APPLY step, AFFIVP is copied into VSF2.SAMPLIB. This program computes the diameter, circumference, and area of a circle. AFFIVP can be edited and printed using TSO commands.

1. Compile and link-edit AFFIVP using the VS FORTRAN Version 2 compiler with the SDUMP option specified explicitly or by default.

   a. Before invoking ISPF/PDF, make sure that "Instructions for ISPF/PDF Users" in topic 5.1 have been completed.

   b. Log on either using the logon procedure or running the CLIST.

   c. Invoke ISPF/PDF by typing the invocation name established at your installation.

   d. When the PRIMARY OPTION MENU appears, select the option that causes the FOREGROUND SELECTION MENU panel to be displayed. This is usually option 4.

   e. Find the line specifying VS FORTRAN Compiler and enter the associated number. (In Figure 15 in topic 5.1.1, option 3 was used.) Enter the data-set name of the program you wish to compile on the next screen. It will probably be VSF2.SAMPLIB(AFFIVP).

      You should receive the following messages from the compilation with no error or warning messages:

```
¦          VS FORTRAN VERSION 2 Release 6 ENTERED.  hh:mm:ss

           **CIRCLE** END OF COMPILATION 1 ****

           **DIAM** END OF COMPILATION 2 ****

           **CIRCUM** END OF COMPILATION 3 ****

           **AREA** END OF COMPILATION 4 ****

¦          VS FORTRAN VERSION 2 Release 6 EXITED.  hh:mm:ss
```

      If you do not receive these messages or you receive error or warning messages, review your installation procedures to this point. Insure that sufficient DASD file space has been allocated. Repeat any steps if necessary. If you cannot identify the source of the problem by reviewing and repeating the procedure, contact your IBM service representative.

   f. Return to the FOREGROUND SELECTION PANEL and find the line specifying linkage editor and enter the associated number. (In Figure 15 in topic 5.1.1, option 7 was used.) For "Other Libraries," specify **VSF2.VSF2FORT** or a name you choose.

2. Run the load module with VS FORTRAN Version 2 interactive debug.

   a. Return to the FOREGROUND SELECTION PANEL.

   b. Find the line specifying VS FORTRAN Version 2 interactive debug, and enter the associated number. (In the example in Figure 15 in topic 5.1.1, option 11 was used.)

   c. The FOREGROUND VS FORTRAN INTERACTIVE DEBUG PANEL will be displayed. Enter the data set on the correct line and specify **AFFIVP** as the member.

   d. Type **DEBUG** opposite the DEBUG option. The next panel displayed should be the INTERACTIVE DEBUG PANEL. If any debuggable program unit's listing is not specified in AFFON, the next panel displayed will be the listings data set specification panel, not the INTERACTIVE DEBUG PANEL. You can enter missing data set definitions on the listings data set specification panel. Press the END key when you are finished. For further information, refer to *VS FORTRAN Version 2 Interactive Debug Guide and Reference*.

To fully verify successful installation of interactive debug, you can issue interactive debug commands as described in "A Sample Interactive Debug Session" in topic 5.4. If you choose not to perform this step, type **QUIT**.

5.2 Instructions for Users of ISPF Without PDF

Proceed only if you are using Interactive System Product Facility (ISPF) Version 2 without ISPF Program Development Facility (PDF). If you are not using ISPF Version 2, go to "Instructions for Non-ISPF Users" in topic 5.3.

To use interactive debug with ISPF without PDF, complete the following steps, as required.

1. Define the necessary data sets for running ISPF and IAD.

2. Modify the ISPF Master Application Menu to include an option for interactive debug, if the menu does not already include this option. The menu may also have to be modified to call an application panel created by your installation (see step 3).

3. Create an application panel that prompts for the name of your load module and then invokes a CLIST created by your installation (see step 4).

4. Create a CLIST to execute the Fortran program and pass the run-time options.

5. Use the interactive debug verification program AFFIVP to verify the availability of Interactive Debug.

Each of these steps is described in more detail in the sections that follow.

Subtopics

5.2.1 1. Defining the Necessary Data Sets

Modify your allocations for the ISPF data sets to include allocations for the CLIST library, the IAD panel library, message library and command table. Examples of lines that could be included in a CLIST are shown in Figure 19.

CAUTION: We advise you to use the procedures in this manual to establish your interactive debug libraries; they are the only ones supported by IAD. If you use another approach, such as the ISPF LIBDEF service, you may get unpredictable results.

```
/*****************************************************************/
/*                                                               */
/* THIS IS AN EXAMPLE OF A CLIST FOR SETTING UP ISPF W/O PDF.  IT */
/* INCLUDES DEFINITIONS FOR THE DATA SETS REQUIRED BY VS FORTRAN  */
/* VERSION 2 INTERACTIVE DEBUG.  THE ISPF NAMES SHOWN IN THIS     */
/* EXAMPLE MAY BE DIFFERENT FROM THE ONES USED AT YOUR LOCATION.  */
/*                                                               */
/*****************************************************************/

FREE  FI(ISPPLIB ISPMLIB ISPLLIB ISPTLIB ISPPROF)
ALLOC FI(ISPPROF)  DA('userid.ISPF.PROFILE') SHR REUSE
ALLOC FI(SYSPROC)  DA('VSF2.VSF2CLIB') SHR REUSE
ALLOC FI(ISPMLIB)  DA('VSF2.VSF2MLIB' 'ISP.V2R2M0.ISPMLIB') SHR REUSE
ALLOC FI(ISPPLIB)  DA('VSF2.VSF2PLIB' 'ISP.V2R2M0.ISPPLIB') SHR REUSE
ALLOC FI(ISPTLIB)  DA('VSF2.VSF2TLIB' 'ISP.V2R2M0.ISPTLIB') SHR REUSE
ALLOC FI(ISPLLIB)  DA('VSF2.VSF2LOAD' 'ISP.V2R2M0.ISPLOAD') SHR REUSE
ALLOC FI(FT05F001) DA(*)
ALLOC FI(FT06F001) DA(*)
```

Figure 19. Sample Modifications to the ISPF Invocation CLIST

5.2.2 2. Modifying the ISPF Master Application Menu

Figure 20 shows how you might modify the ISPF master application menu to provide an option for interactive debug (option 2 in this example) and invoke a prompt panel (called USERISP in this example).

```
%------------------------  ISPF MASTER APPLICATION MENU  -----------------------
%OPTION  ===>_ZCMD                           %     +USERID   - &ZUSER
%   1 +SAMPLE1     - Sample application 1          +TIME     - &ZTIME
%   2 +VSF IAD     - VS FORTRAN Interactive Debug  +TERMINAL - &ZTERM      .
%   3 +.           - (Description for option 3)    +PF KEYS  - &ZKEYS
%   4 +.           - (Description for option 4)
%   5 +.           - (Description for option 5)
%   X +EXIT        - Terminate ISPF using list/log defaults
%
+Enter%END+command to terminate ISPF.
%
)INIT
  .HELP    = ISP00005     /* Help for this master menu            */
  &ZPRIM   = YES          /* This is a primary option menu        */
)PROC
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
                 1,'PANEL(ISP@PRIM)' /* Sample primary option menu */
                 2,'PANEL(USERISP) NEWAPPL(AFF)'  /* SAMPLE PANEL TO INVOKE IAD */
              /*******************************************************/
              /*                                                     */
              /* Add other applications here.                        */
              /*                                                     */
              /*******************************************************/
                ' ',' '
                 X,'EXIT'
                 *,'?' )
)END
```

Figure 20. Sample Modifications to ISPF Master Application Menu

5.2.3 3. Creating an Application Panel

Create an application panel at your installation similar to the one shown in Figure 21. The sample prompts the user for the name of the text file, and then invokes a CLIST called USERDBG.

```
)ATTR DEFAULT(%+_)
      /*  % TYPE(TEXT) INTENS(HIGH)      defaults displayed for     */
      /*  + TYPE(TEXT) INTENS(LOW)       information only           */
      /*  _ TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)            */
  ! TYPE(INPUT) INTENS(LOW) PAD(' ')  /* input field padded with '' */
)BODY
%--------------------  SAMPLE VSF IAD PANEL -----------------------------------
%COMMAND ===>_ZCMD                                                        +
%
+   ENTER THE NAME OF YOUR PROGRAM BELOW...
+  %===>!MEM
+
+   ENTER THE NAME OF YOUR LIBRARY BELOW...
+  %===>!LIB

+   ENTER THE LIST OF YOUR EXECUTION TIME OPTIONS BELOW...
+  %===>!FDEBUG
+
)PROC
 VER(&MEM,NB,NAME)
 VPUT (MEM,LIB,FDEBUG) PROFILE
 &ZSEL='CMD(%USERDBG)'
)END
```

Figure 21. Sample Application Panel (USERISP) to Prompt for the Text File Name

5.2.4 4. Creating a CLIST to Pass the Run-Time Options

The sample CLIST in Figure 22 invokes AFFLOAD, which in turn loads the Fortran program with the DEBUG parameter. For this example, we have called the CLIST "USERDBG."

```
------------------------------------------------------------------------

  PROC 0
  CONTROL NOLIST MAIN NOFLUSH NOMSG
  /*    MEM    - MEMBER  NAME                                          */
  /*    LIB    - LIBRARY NAME                                          */
  /*    FDEBUG - EXECUTION TIME OPTIONS                                */
  ISPEXEC VGET (MEM,LIB,FDEBUG,ZPREFIX)
  SET &FAFFID = &MEM                        /* DEFAULT TO MEMBER NAME  */
  SET &ZFAFFIN = &STR('&ZPREFIX..&FAFFID..RESTART')
  SET &ZFAFFOU = &STR('&ZPREFIX..&FAFFID..LOG')
  SET &ZFAFFON = &STR('&ZPREFIX..&FAFFID..INCLUDE')
  SET &ZFAFFPR = &STR('&ZPREFIX..&FAFFID..PRINT')
  FREE FI(AFFPRINT AFFIN AFFOUT AFFON AFFLOAD)
  ALLOC FI(AFFIN) DA(&ZFAFFIN) SHR
  IF &LASTCC ¬= 0 THEN ALLOC FI(AFFIN) DUMMY RECFM(F)
  ALLOC FI(AFFON) DA(&ZFAFFON) SHR
  IF &LASTCC ¬= 0 THEN ALLOC FI(AFFON) DUMMY RECFM(F)
  DELETE &ZFAFFOU
  ALLOC FI(AFFOUT) DA(&ZFAFFOU) MOD CATALOG SPACE (1) CYLINDERS
  DELETE &ZFAFFPR
  ALLOC FI(AFFPRINT) DA(&ZFAFFPR) MOD CATALOG SPACE(1) CYLINDERS
  ALLOC FI(AFFLOAD) DA(&LIB) SHR
  ISPEXEC SELECT PGM(AFFLINKF) PARM(&MEM/&FDEBUG) NEWAPPL(AFF) NEWPOOL
  FREE FI(AFFPRINT AFFIN AFFOUT AFFON AFFLOAD)


------------------------------------------------------------------------
```

Figure 22. Sample CLIST (USERDBG) to Invoke the Program Without PDF

5.2.5 5. Verifying Availability of Interactive Debug

Use the interactive debug verification program AFFIVP provided on the product tape. During the APPLY step, AFFIVP is copied into VSF2.SAMPLIB. This program computes the diameter, circumference and area of a circle. AFFIVP can be edited and printed using TSO commands.

1. Compile and link-edit AFFIVP using the VS FORTRAN Version 2 compiler with the SDUMP option specified explicitly or by default. Figure 23 shows the sample JCL that runs the compile-link procedure, VSF2CL.

   Note: If you did not use the IBM-supplied jobs, you will need to change the VSF2 data set prefix to the appropriate name for your site.

```
-------------------------------------------------------------------------

  //AFFIVP      JOB . . .
  //FORTRAN     EXEC VSF2CL
  //FORT.SYSIN  DD DSN=VSF2.SAMPLIB(AFFIVP),DISP=SHR


-------------------------------------------------------------------------
```

Figure 23. Job to Verify Successful Preparation to Use Interactive Debug

   During the APPLY step of installation, VSF2CL is copied into VSF2.PROCLIB and AFFIVP is copied into VSF2.SAMPLIB. To run VSF2CL, you must do one of the following:

   Put it in a system PROCLIB, or

   Copy it into the AFFIVP job after the JOB statement and before the EXEC statement. Remember to add ***//* PEND** after the last line of the procedure.

   Note: The procedure VSF2CL uses a prefix of SYS1 for the VS FORTRAN data sets. If you used the IBM-supplied jobs, these data sets will have a prefix of VSF2; therefore, you may need to edit the VSF2CL procedure.

   You should receive the following messages from the compilation with no error or warning messages:

```
¦      VS FORTRAN VERSION 2 Release 6 ENTERED.  hh:mm:ss

       **CIRCLE** END OF COMPILATION 1 ****

       **DIAM** END OF COMPILATION 2 ****

       **CIRCUM** END OF COMPILATION 3 ****

       **AREA** END OF COMPILATION 4 ****

¦      VS FORTRAN VERSION 2 Release 6 EXITED.  hh:mm:ss
```

   If you do not receive these messages or you receive error or warning messages, review your installation procedures to this point. Insure that sufficient DASD file space has been allocated. Repeat any steps if necessary. If you cannot identify the source of the problem by reviewing and repeating the procedure, contact your IBM service representative. Remember to save the compilation listing for use with interactive debug.

2. Run the load module with VS FORTRAN Version 2 interactive debug.

   a. Before invoking ISPF, make sure that "Instructions for Users of ISPF Without PDF" in topic 5.2 have been completed.

   b. Invoke ISPF by typing the name established at your installation.

c. When the ISPF MASTER APPLICATION MENU appears, select the VS FORTRAN interactive debug option, usually option 2.

d. Type **AFFIVP** on the line labeled FILE ID, and type **DEBUG** opposite **OPTIONS ===>**. The next panel displayed should be the INTERACTIVE DEBUG PANEL. However, if any debuggable program unit's listing is not specified in the AFFON file, the next panel displayed will be the listings data set specification panel, not the INTERACTIVE DEBUG PANEL. You can enter missing data set definitions on the listings data set specification panel. Press END when you are done. For further information, refer to *VS FORTRAN Version 2 Interactive Debug Guide and Reference*.

To fully verify successful installation of interactive debug, you can now issue interactive debug commands as described in "A Sample Interactive Debug Session" in topic 5.4. If you choose not to perform this step, type **QUIT**.

5.3 Instructions for Non-ISPF Users

Subtopics

5.3.1 1. Running Interactive Debug in Line Mode

Do this in *one* of the following ways:

Include a STEPLIB DD statement for VSF2LOAD (the load library containing the interactive debug load module) in your TSO logon procedure.

Include VSF2LOAD in the LNKLST concatenation

If your load module was link-edited using the Version 2 Release 3 (or later) Library, allocate VSF2.VSF2LOAD to the ddname FORTLIB:

```
ALLOCATE DDNAME(FORTLIB) DSN('VSF2.VSF2LOAD') SHR REUSE
```

If you installed the interactive debug HELP member in a library other than SYS1.HELP, you must concatenate that library's name to the SYSHELP DD statement in your TSO logon procedure.

Note: The defaults of the IBM-supplied jobs will put the interactive debug HELP member into VSF2.HELP.

5.3.2 2. Verifying Availability of Interactive Debug

Use the interactive debug verification program AFFIVP provided on the product tape. During the APPLY step, AFFIVP is copied into VSF2.SAMPLIB. This program computes the diameter, circumference and area of a circle. AFFIVP can be edited and printed using TSO commands.

1. Compile and link-edit AFFIVP using the VS FORTRAN Version 2 compiler with the SDUMP option specified explicitly or by default. Figure 24 shows the sample JCL that runs the compile-link procedure, VSF2CL.

   Note: If you did not use the IBM-supplied jobs, you will need to change the VSF2 data set prefix to the appropriate name for your site.

```
-------------------------------------------------------------------------

  //AFFIVP      JOB . . .
  //FORTRAN     EXEC VSF2CL
  //FORT.SYSIN  DD DSN=VSF2.SAMPLIB(AFFIVP),DISP=SHR


-------------------------------------------------------------------------
```

Figure 24. Job to Verify Successful Preparation to Use Interactive Debug

   During the APPLY step of installation, VSF2CL is copied into VSF2.PROCLIB and AFFIVP is copied into VSF2.SAMPLIB. To run VSF2CL, you must either:

   Put it in a system PROCLIB, or

   Copy it into the AFFIVP job after the JOB statement and before the EXEC statement. Remember to add ***//* PEND** after the last line of the procedure.

   Note: The procedure VSF2CL uses a prefix of SYS1 for the VS FORTRAN data sets. If you used the IBM-supplied jobs, these data sets will have a prefix of VSF2; therefore, you may need to edit the VSF2CL procedure.

   You should receive the following messages from the compilation with no error or warning messages:

```
¦      VS FORTRAN VERSION 2 Release 6 ENTERED.  hh:mm:ss

       **CIRCLE** END OF COMPILATION 1 ****

       **DIAM** END OF COMPILATION 2 ****

       **CIRCUM** END OF COMPILATION 3 ****

       **AREA** END OF COMPILATION 4 ****

¦      VS FORTRAN VERSION 2 Release 6 EXITED.  hh:mm:ss
```

   If you do not receive these messages or you receive error or warning messages, review your installation procedures to this point. Insure that sufficient DASD file space has been allocated. Repeat any steps if necessary. If you cannot identify the source of the problem by reviewing and repeating the procedure, contact your IBM service representative.

2. After AFFIVP has been compiled and link-edited into a load module, run the load module with VS FORTRAN Version 2 interactive debug.

   a. Use an editor to build a CLIST to run a Fortran program with interactive debug. (See Figure 25.) The CLIST must include ALLOCATE statements for all data sets used by VS FORTRAN Version 2 interactive debug. The CLIST shown in Figure 25 has three parameters:

MEMBER The member to be run; has to be a load module that is ready to be run.

DSN The data set containing that member; default is FORTRAN.LOAD.

OPTION The run-time option; default is DEBUG.

b. Invoke the CLIST and the debug program AFFIVP by entering:

```
clistname AFFIVP
```

where **clistname** is replaced by the name you used for the CLIST.

```
PROC 1 MEMBER DSN(FORTRAN.LOAD) OPTION(DEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOSYMLIST NOCONLIST
IF &OPTION = DEBUG THEN DO
   FREE FI(AFFPRINT AFFON AFFIN)
   ALLOC FI(AFFIN) DA(&MEMBER..RESTART) SHR
   IF &LASTCC ¬= 0 THEN ALLOC FI(AFFIN) DUMMY
   ALLOC FI(AFFON) DA(&MEMBER..INCLUDE) SHR
   IF &LASTCC ¬= 0 THEN ALLOC FI(AFFON) DUMMY
   ALLOC FI(AFFPRINT) DA(&MEMBER..PRINT) SHR
   IF &LASTCC ¬= 0 THEN +
      ALLOC FI(AFFPRINT) DA(&MEMBER..PRINT) +
      NEW CATALOG SPACE(5 5) TRACKS
END
CALL '&SYSUID..&DSN.(&MEMBER)' '&OPTION'
SET RCODE = &LASTCC
FREE FI(AFFPRINT AFFON)
WRITE RETURN CODE: &RCODE
EXIT
END
```

Figure 25. TSO CLIST to Invoke a VS FORTRAN Version 2 Program

If your CLIST runs properly, you will receive an informational message and several lines of copyright information, followed by a WHERE message, identifying the statement about to be run. This is followed by the interactive debug prompt, **FORTIAD**, indicating that the VS FORTRAN program has reached the first debugging hook. For example:

```
VS FORTRAN VERSION 2 RELEASE 5 INTERACTIVE DEBUG
5668-806 (C) COPYRIGHT IBM CORP. 1985, 1990
LICENSED MATERIALS-PROPERTY OF IBM
WHERE: CIRCLE.7
FORTIAD
```

To fully verify successful installation of interactive debug, you can issue interactive debug commands, as described in the following section. If you choose not to perform this step, type **QUIT**.

5.4 A Sample Interactive Debug Session

The interactive debug verification program AFFIVP will generate input/output as shown in Figure 26. You can enter the commands and data shown in this example to verify successful installation of interactive debug. You can also enter other interactive debug commands, if you choose. In the figure, all lines beginning with "**=***" indicate commands or data entered on the terminal. However, when you enter the commands, do not type the preceding "**=***" symbols. (The FORTIAD prompt does not appear since this log was obtained during running under ISPF/PDF.) In line mode (for non-ISPF users), the initial equal signs (=) do not appear.

If you are using an AFFON file, you will receive information indicating whether or not AFFON processed successfully.

```
  =VS FORTRAN VERSION 2 RELEASE 5 INTERACTIVE DEBUG
  =5668-806 (C) COPYRIGHT IBM CORP. 1985, 1990
  =LICENSED MATERIALS-PROPERTY OF IBM
  =WHERE: CIRCLE.7
  =* listsubs
  =PROGRAM UNIT                     COMPILER   OPT   HOOKED TIMING
  =CIRCLE                          VSF 2.5.0    0      YES    OFF
  =DIAM                            VSF 2.5.0    0      YES    OFF
  =CIRCUM                          VSF 2.5.0    2      YES    OFF
  =AREA                            VSF 2.5.0    3      YES    OFF
  =* describe (data pi)
  =CIRCLE.DATA:                     REAL*4
  =   RANK = 1,  SIZE = 3 ELEMENTS
  =   DIM 1:    EXTENT = 3   LBOUND (1),  UBOUND (3)
  =CIRCLE.PI:                       REAL*8
  =* at diam.entry (step)
  =* go
  =FT06F001  ENTER THE VALUE OF THE CIRCLE RADIUS (xxx.xx):
  =FT05F001 INPUT: PRECEDE INPUT WITH % OR ENTER IAD COMMAND
  =* %352.67
  =AT: DIAM.ENTRY
  =NEXT: DIAM.3
  =* set diam.value = 0.0
  =* when test value
  =* go
  =WHEN: "TEST" SATISFIED;
  =CURRENTLY AT DIAM.4
  =* offwn test
  =* at circle.42 (list '= READY FOR TERMINATION ='%go)
  =* listbrks
  =CURRENT BREAKPOINTS:
  =  CIRCLE.42
  =  DIAM.ENTRY
  =CURRENT WHEN CONDITIONS:
  =  TEST  OFF   DIAM.VALUE
  =CURRENT HALT STATUS: OFF
  =* go
  =FT06F001  THE DIAMETER OF THE CIRCLE IS  705.34
  =FT06F001  THE CIRCUMFERENCE OF THE CIRCLE IS 2215.89
  =FT06F001  THE AREA OF THE CIRCLE IS  390738.94
  =AT: CIRCLE.42
  == READY FOR TERMINATION =
  =PROGRAM HAS TERMINATED; RC ( 0)
  =* quit
```

Figure 26. TSO Interactive Debug Input/Output

6.0 Chapter 6. Customizing VS FORTRAN Version 2

This is an optional step. If you wish to proceed, make sure you have completed all steps under Chapter 4, "Installing VS FORTRAN Version 2" in topic 4.0.

Note: If you are customizing VS FORTRAN Version 2 under MVS/XA or MVS/ESA, you must have Assembler H Version 2 available.

We recommend that you accept VS FORTRAN Version 2 before doing any customization on the product.

Subtopics

6.1 Customizing Cataloged Procedures

After installation, the following VS FORTRAN Version 2 cataloged procedures are placed in the procedure library VSF2.PROCLIB. Refer to *VS FORTRAN Version 2 Programming Guide* for more information.

VSF2C Compiles a source program

VSF2CL Compiles a source program and link-edits the object module into a load module

VSF2CG Compiles a source program and loads and runs the object module

VSF2CLG Compiles a source program, link-edits the object module into a load module, and runs the load module

VSF2L Loads and runs an object module with the loader program

VSF2LG Link-edits an object module into a load module and runs the load module

VSF2G Runs a link-edited object module

VFT2RCL Compiles a reentrant source program and link-edits the object module into a load module

VFT2RCLG Compiles a reentrant source program, link-edits the object module into a load module, and runs the load module

VFT2RLG Link-edits a reentrant object module into a load module and runs it

You can customize the cataloged procedures in the following ways:

Increase FVREGN in the PROC statement

If you changed the names of any of the VS FORTRAN Version 2 data sets, change the corresponding data set name (DSN) parameters on the STEPLIB and SYSLIB DD statements. The procedures assume a prefix of SYS1; however, the IBM-supplied installation jobs use VSF2.

In the linkage editor (LKED) step, the SYSLIB DD statement specifies run-time library routines in load mode

```
//SYSLIB DD DSN=SYS1.VSF2FORT,DISP=SHR
```

To specify link mode, change the SYSLIB DD statement as follows:

```
//SYSLIB DD DSN=SYS1.VSF2LINK,DISP=SHR
//       DD DSN=SYS1.VSF2FORT,DISP=SHR
```

See "Specifying the Mode of Run-Time Library Modules" in topic 6.5 for more information.

To make the alternative math library routines available, see the section "Making the Alternative Math Library Subroutines Available" in topic 6.2.

If you are not using interactive debug *and* wish to specify run-time routines in link mode, delete the following statement:

```
//STEPLIB DD DSN=SYS1.VSF2LOAD,DISP=SHR
```

6.2 Making the Alternative Math Library Subroutines Available

The alternative math library contains the old standard math subroutines. These subroutines are link-edited in VSF2MATH by the installation process.

In order to make the alternative math routines available to all users, you must change the cataloged procedures to concatenate VSF2MATH ahead of VSF2FORT in the link-edit step SYSLIB DD statement. The installation jobs assume a prefix of VSF2 whereas the cataloged procedures assume a prefix of SYS1. Cataloged procedures, such as VSF2CL and VSF2CLG, are provided in PROCLIB. For example, use these statements in load mode:

```
//SYSLIB DD DSN=SYS1.VSF2MATH,DISP=SHR
//       DD DSN=SYS1.VSF2FORT,DISP=SHR
```

or use these statements in link mode:

```
//SYSLIB DD DSN=SYS1.VSF2MATH,DISP=SHR
//       DD DSN=SYS1.VSF2LINK,DISP=SHR
//       DD DSN=SYS1.VSF2FORT,DISP=SHR
```

Note: The VS FORTRAN Version 1 standard math routines (VSF2MATH) are not supported for parallel or vector programs.

6.3 Making Load Module AFBVRENT Available

You may have load modules that were created at VS FORTRAN Version 1 Release levels 2, 3, or 3.1. If you do, and these modules were link-edited so that they use the reentrant I/O library load module IFYVRENT, they cannot be run until you make AFBVRENT available. To correct this problem, you may do one of the following:

In any JCL that runs these programs, add a STEPLIB or JOBLIB DD statement that refers to VSF2.VSF2LOAD

You may choose to put the module AFBVRENT and its alias IFYVRENT in the pageable link pack area (LPA) by moving it to SYS1.LPALIB. In an MVS/XA or MVS/ESA environment, this module will reside below the 16-megabyte virtual storage line.

If you have load modules that were created before VS FORTRAN Version 1 Release 2 and they used the reentrant I/O library load module IFYVRENT, be aware that they are not compatible with the reentrant I/O library load module AFBVRENT. **You must link-edit these modules again using the VS FORTRAN Version 2 library.**

6.4 Rebuilding Composite Modules

After running the ACCEPT job in the installation process, you can run the jobs shown in Figure 27 in topic 6.4.1 and Figure 28 in topic 6.4.2 to rebuild composite modules AFBVRENA, AFBVRENB, and AFBVRENC. The jobs are in the AFBLBS data set.

The composite module AFBVRENP cannot be customized.

Jobs for deleting or adding optional modules are described below. To delete and add both, use the appropriate delete job first, then the add job.

Subtopics

6.4.1 Deleting Optional Modules from a Composite Module

The jobs in Figure 27 each contain inline SMP/E UCLIN and LINK-EDIT JCL to delete optional modules you do not want included in your composite modules.

Figure 27. Jobs to Delete Optional Modules from a Composite Module

Job Name	**Composite Module**
AFBDRENA	AFBVRENA
AFBDRENB	AFBVRENB
AFBDRENC	AFBVRENC

The UCLIN step tells SMP/E to delete optional modules that you do not want to include in your tailored composite modules. Thus, you must remove the DEL statements of any optional module you want to include in your composite module. If you run any of the jobs shown in Figure 27 without modifying them, you receive a composite module containing only the required modules.

The LINK-EDIT step performs the actual link-edit of the tailored composite module by replacing the modules you have specified. The REPLACE statements you keep in the LINK-EDIT step must match the modules you specified in the UCLIN step.

Be certain that you retain the INCLUDE statement for the original composite module.

6.4.2 Adding Optional Modules to a Composite Module

The jobs in Figure 28 contain inline SMP/E UCLIN and LINK-EDIT JCL to add optional modules you want to include in your tailored composite modules.

Figure 28. Jobs to Add Optional Modules to a Composite Module

Job Name	**Composite Module**
AFBARENA	AFBVRENA
AFBARENB	AFBVRENB
AFBARENC	AFBVRENC

The UCLIN step tells SMP/E to add the optional modules you want to include in your tailored composite module. You must remove the ADD statement for each optional module you do not want to add to your tailored composite module. If you run any of the jobs shown in Figure 28 without first modifying them, you receive a composite module containing all the required and optional modules.

Note: If you attempt to add a module that is already in the composite module, you receive an SMP/E error message.

The LINK-EDIT step performs the actual link-edit of the tailored composite module by including the routines you have specified. The INCLUDE statements you keep in the LINK-EDIT step must match the routines you want to include in your tailored composite module, regardless of whether you are adding the routine in the UCLIN step above or it was already in the composite module. INCLUDE statements are not provided for any of the **required** modules.

Be certain that you retain the INCLUDE statement for the original composite module.

6.4.3 Placing Composite Modules in Link Pack Areas

After you have modified the jobs, you can run each job. In each case, the old copy of the composite module will be replaced with a rebuilt copy in both the VSF2.VSF2FORT and VSF2.VSF2LOAD libraries. In addition you can:

- For MVS/SP1 systems, place AFBVRENC in the link pack area for shared use by all regions. If it is not placed in the link pack area, it is loaded from library VSF2.VSF2LOAD.

- For MVS/XA and ESA systems, place AFBVRENA in the extended link pack area (ELPA) for shared use by all regions. If it is not placed in the link pack area, it is loaded from library VSF2.VSF2LOAD. AFBVRENA must have a residence mode of ANY so that it can reside above the 16-megabyte virtual storage line.

- For MVS/XA and ESA systems, place AFBVRENB in the link pack area for shared use by all regions. If it is not placed in the link pack area, it is loaded from the library VSF2.VSF2LOAD.

  AFBVRENB must have a residence mode of 24 so that it resides below the 16-megabyte virtual storage line.

For additional information, see Appendix B, "Composite Modules" in topic B.0.

6.5 Specifying the Mode of Run-Time Library Modules

Programmers can choose to combine run-time libraries (other than the math routines) with their programs in either **link mode** or **load mode**. Only **load mode** is supported for a parallel program. MTF programs that use link mode must be converted to **load mode** if they are to be run in parallel.

In **link mode**, library modules are included as part of the program, along with compiler-generated code.

In **load mode**, library modules are loaded dynamically when the program is started. This run-time loading has several advantages:

- Allows application programs to run above the 16-megabyte line
- Reduces auxiliary storage requirements for load modules
- Speeds link-editing
- Allows some library routines to be placed in the extended link pack area in an MVS/XA or MVS/ESA environment.

For the best run-time performance, run programs in **load mode** under these conditions:

- Put all modules that may be used by your applications in composite modules (see Appendix B, "Composite Modules" in topic B.0)
- Place the composite modules in the link pack area
- Run multiple Fortran programs in different address spaces.

After installation of the VS FORTRAN Version 2 complete or compiler and library product, you may wish to customize the cataloged procedures (for example, VSF2CLG for compile, link-edit, and go), to specify the libraries in load mode or link mode. All cataloged procedures provided with the product are set up for load mode. The methods for specifying libraries in link mode or load mode are described below.

Subtopics
6.5.1 Specifying Link Mode
6.5.2 Specifying Load Mode

6.5.1 Specifying Link Mode

For operation in link mode, concatenate VSF2LINK ahead of VSF2FORT for use by the linkage editor when it includes VS FORTRAN Version 2 library modules. Specify both VSF2LINK and VSF2FORT in the DD statement for SYSLIB in the linkage editor step:

```
//SYSLIB  DD  DSN=VSF2.VSF2LINK,DISP=SHR
//        DD  DSN=VSF2.VSF2FORT,DISP=SHR
```

A program link-edited in link mode does not require specification of any VS FORTRAN Version 2 libraries at run time. However, there are two exceptions:

1. If your program is reentrant, specify the library containing the load module with the nonshareable parts of the reentrant program.

2. If you are using VS FORTRAN Version 2 interactive debug, specify VSF2FORT in STEPLIB, JOBLIB, or LNKLST.

6.5.2 Specifying Load Mode

Note: VSF2FORT may be used in place of VSF2LOAD in this section.

For operation in load mode, specify only VSF2.VSF2FORT in the DD statement for SYSLIB in the linkage editor step:

```
//SYSLIB  DD  DSN=VSF2.VSF2FORT,DISP=SHR
```

In order for a program that has been link-edited in load mode to be run, VSF2LOAD must be made available for the run step by performing one of the following:

Concatenate VSF2.VSF2LOAD to SYS1.LINKLIB in the system link list so that VSF2.VSF2LOAD is searched as part of the link library without JOBLIB or STEPLIB DD statements.

Place the reentrant composite modules AFBVRENA (MVS/XA or MVS/ESA), AFBVRENB (MVS/XA or MVS/ESA), AFBVRENC (MVS/SP Version 1) and AFBVRENP, as well as selected individual reentrant modules, in the link pack area (SYS1.LPALIB).

Note: This step must be repeated manually each time service is applied.

Use the copy of the modules in the link pack area without searching VSF2.VSF2LOAD. (If maintenance affects any modules in the link pack area, copy the updated copies of the modules into the link pack area from VSF2.VSF2LOAD.)

To use a step library or job library in addition to loading reentrant modules from the link pack area, do the following

1. After modifying the composite modules, place the reentrant composite modules AFBVRENA (MVS/XA or MVS/ESA), AFBVRENB (MVS/XA or MVS/ESA), AFBVRENC (MVS/SP Version 1) and AFBVRENP in the link pack area (library SYS1.LPALIB).

   Note: This step must be repeated manually each time service is applied.

2. Optionally, place any reentrant modules that are *not* in a composite module into the link pack area.

3. Create a new library that contains all modules from VSF2.VSF2LOAD *minus* the modules (either composite modules or individual modules) that have been placed in the link pack area. Make this library available as either a step library or as a job library for the running of the VS FORTRAN Version 2 program.

If maintenance affects any of the modules in the link pack area or your new library, copy the updated copies of the modules from VSF2.VSF2LOAD.

At run time, the application programmer can place the following JOBLIB DD statement in the JCL for the job that runs the VS FORTRAN Version 2 program:

```
//JOBLIB   DD  DSN=VSF2.VSF2LOAD,DISP=SHR
```

Alternatively, the programmer can place the following STEPLIB DD statement in the JCL for the step that runs the VS FORTRAN Version 2 program:

```
//STEPLIB  DD  DSN=VSF2.VSF2LOAD,DISP=SHR
```

This technique does not let you use reentrant modules that are in the link pack area, because step libraries and job libraries are searched before the link pack area. (Refer to *MVS/SP Supervisor Services and Macro Instructions*, or *MVS/XA Supervisor Services and Macro Instructions*, for a discussion of program management.)

6.6 Changing Option Defaults

After successfully accepting VS FORTRAN Version 2, the jobs for customizing the compiler may be found in the ILXCCS data set, and the jobs for customizing the run-time environment may be found in the AFBLBS data set. They are shown in Figure 29.

Figure 29. Customization Jobs

Job Name	Function	Procedure Invoked
ILXCOME	Customize compiler	**aaa**EPROC
AFBUNTE	Customize unit attribute table	**aaa**EPROC
AFBEXE	Customize run-time options	**aaa**EPROC
AFBERRE	Customize error option table	**aaa**EPROC

In this table, **aaa** is **AFF** if you are installing the complete product, **ILX** if you are installing the compiler and library product, and **AFB** if you are installing the library-only product.

These jobs are provided for your convenience. They currently include the IBM-supplied defaults. You may want to customize these jobs for your location by modifying the appropriate parameters.

You may prefer to use your own procedures for receiving, applying, and accepting the USERMODS. If so, the following SYSLIB information may be helpful:

```
//SYSLIB    DD   DSN=&SMPPRFX..SMPMTS,DISP=SHR
            DD   DSN=&DISPRFX..ILXCCS,DISP=SHR
            DD   DSN=&DISPRFX..AFBLBS,DISP=SHR
            DD   DSN=&DISPRFX..AFFSRC,DISP=SHR
```

Note: For the library-only, delete the DD statements for ILXCCS and AFFSRC. For the compiler and library product, delete the DD statement for AFFSRC.

For guidelines on invoking macros, refer to "Guidelines for Invoking Macros" in topic A.1.

Subtopics

6.6.1 Changing Compile-Time Option Defaults

Note: If you installed the library-only, skip this section.

For additional information on the VSF2COM macro, refer to page "VSF2COM: Changes Compile-Time Option Defaults" in topic A.2.

The product tape provides a module ILX0OPTS, which contains a set of defaults for compile-time options. To customize ILX0OPTS, you can use the IBM-supplied job ILXCOME, or you can code a VSF2COM macro instruction in an SMP/E USERMOD.

6.6.2 Changing the Unit Attribute Table Defaults

For additional information on macros VSF2UAT, VSF2UNIT, and VSF2DCB, refer to Appendix A, "Customization Macros" in topic A.0.

CAUTION: If you change the IBM-supplied default DCB values, the existing Fortran programs that depend on the original defaults may not work. For more information on DCB values, refer to *VS FORTRAN Version 2 Programming Guide*.

The product tape provides a module AFBVUAT, the Unit Attribute Table, which contains a set of defaults and DCB information for each I/O unit. To customize AFBVUAT for your installation, you may use the IBM-supplied job AFBUNTE, or you may code VSF2UAT, VSF2UNIT, and VSF2DCB macro instructions in an SMP/E USERMOD.

IBM-Supplied Default Values: The macro instructions shown in Figure 30 are provided in the module AFBVUAT. This module is used to set up the IBM-supplied default values for the standard I/O units, and file characteristics such as the DCB information. Note that the last VSF2DCB macro does not have a label; its set of defaults apply to all units except 5, 6, and 7.

```
------------------------------------------------------------------------

  AFBVUAT VSF2UAT UNTABLE=99,
                  DECIMAL=PERIOD,
                  READER=5,
                  ERRMSG=6,
                  PRINTER=6,
                  PUNCH=7,
                  DEVICE=SYSDA

         VSF2UNIT  5,DCBSET=DCBRDR
         VSF2UNIT  6,DCBSET=DCBPRT
         VSF2UNIT  7,DCBSET=DCBPUN

  DCBRDR VSF2DCB   SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                   SURECFM=F,SULRECL=80,SUBLKSI=80

  DCBPRT VSF2DCB   SFRECFM=UA,SFLRECL=133,SFBLKSI=133

  DCBPUN VSF2DCB   SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                   SURECFM=F,SULRECL=80,SUBLKSI=80

          VSF2DCB  SFRECFM=U,SFLRECL=800,SFBLKSI=800,SFMAXRE=100,
                   SURECFM=VS,SULRECL=-1,SUBLKSI=800,SUMAXRE=100,
                   DMAXRE=100

          VSF2UAT  TYPE=FINAL


------------------------------------------------------------------------
Figure 30. IBM-Supplied Macro Instructions
```

Note: The above format is given for readability purposes. However, if you want to insert it into a USERMOD, remember to add the necessary continuation flags in column 72, and to begin continued lines in column 16.

Examples of Changing Default Values: The following examples show how you could modify the IBM-supplied defaults for your own environment. You can alter instructions by striking over existing data or you can add more VSF2UNIT and VSF2DCB macro instructions.

Example 1:

In this example, we have specified a device named SYSSQ as a DASD file and assigned a unique set of DCB attributes to units 1 through 4. The DCB Information for both sequential formatted and unformatted files is indicated in the first VSF2DCB macro ("USERDCB") shown in Figure 31.

```
------------------------------------------------------------------------

  AFBVUAT   VSF2UAT    DEVICE=SYSSQ
            VSF2UNIT   (1,4),DCBSET=USERDCB
            VSF2UNIT   5,DCBSET=DCBRDR
            VSF2UNIT   6,DCBSET=DCBPRT
            VSF2UNIT   7,DCBSET=DCBPUN

  USERDCB   VSF2DCB    SFRECFM=FB,SFLRECL=50,SFBLKSI=250,SFMAXRE=200,
                       SURECFM=FB,SULRECL=50,SUBLKSI=250,SUMAXRE=200,
                       DMAXRE=200

  DCBRDR    VSF2DCB    SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                       SURECFM=F,SULRECL=80,SUBLKSI=80

  DCBPRT    VSF2DCB    SFRECFM=UA,SFLRECL=133,SFBLKSI=133

  DCBPUN    VSF2DCB    SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                       SURECFM=F,SULRECL=80,SUBLKSI=80

            VSF2DCB    SFRECFM=U,SFLRECL=800,SFBLKSI=800,SFMAXRE=100,
                       SURECFM=VS,SULRECL=-1,SUBLKSI=800,SUMAXRE=100,
                       DMAXRE=100

            VSF2UAT    TYPE=FINAL


------------------------------------------------------------------------
Figure 31. Modified IBM-Supplied Macro Instructions
```

Note: The above format is given for readability purposes. However, if you want to insert it into a USERMOD, remember to add the necessary continuation flags in column 72, and to begin continued lines in column 16.

VSF2UAT, VSF2UNIT, and VSF2DCB must all be coded, in that order, followed by the TYPE=FINAL statement.

Example 2:

If you want to change the default standard I/O unit values for **READER**, **ERRMSG**, **PRINTER**, and **PUNCH** to 1, 2, 3, 4, respectively, modify the IBM-supplied macros as shown in Figure 32.

```
------------------------------------------------------------------------

  AFBVUAT   VSF2UAT    DECIMAL=PERIOD,
                       READER=1,
                       ERRMSG=2,
                       PRINTER=3,
                       PUNCH=4,
                       DEVICE=SYSDA

            VSF2UNIT   1,DCBSET=DCBRDR
            VSF2UNIT   2,DCBSET=DCBTERM
            VSF2UNIT   3,DCBSET=DCBPRT
            VSF2UNIT   4,DCB=DCBPUN
```

.

```
.
.
```

Figure 32. Modified IBM-Supplied Macro Instructions

6.6.3 Changing Run-Time Option Defaults

The product tape provides a module AFBVGPRM, which contains a set of defaults for run-time options. To customize AFBVGPRM, you may use the IBM-supplied job AFBEXE, or you may code a VSF2PARM macro instruction in an SMP/E USERMOD.

For more VSF2PARM macro information, see "VSF2PARM: Changes Run-Time Option Defaults" in topic A.6.

6.6.4 Customizing the Error Option Table

The product tape provides a module AFBUOPT, which is an error option table that specifies the following items for each run-time error defined by VS FORTRAN Version 2:

The number of times the error is allowed to occur before the user's program terminate

The maximum number of times the message may be printe

Whether or not the traceback map is to be printed with the messag

Whether or not a user-written error exit routine is called

To customize AFBUOPT, you may use the IBM-supplied job AFBERRE, or you may code your own VSF2UOPT macro instruction in an SMP/E USERMOD.

Note: Customization applies to the permanent copy of the error option table in the VS FORTRAN Version 2 library. Each individual Fortran program can also make run-time changes to its copy of the table by calling the supplied subroutines ERRSET, ERRSAV, and ERRSTR. See *VS FORTRAN Version 2 Language and Library Reference* and *VS FORTRAN Version 2 Programming Guide* for more information.

For additional VSF2UOPT macro information, see "VSF2UOPT: Customizes the Error Option Table" in topic A.7.

6.7 Installing the Compiler in the Link Pack Area

The VS FORTRAN Version 2 compiler consists of three load modules:

FORTVS
ILX0FOR
ILX0TRC

Each load module can be installed in the link pack area (LPA).

The ILX0FORT module has the attribute RMODE ANY on MVS/XA or MVS/ESA, and can be installed in the extended link pack area (ELPA). Installing ILX0FORT in the ELPA:

1. Frees approximately 1.8 megabytes of storage within the 16MB area for other uses.

2. Reduces paging overhead because MVS uses a single copy of each page for all users performing compilations, rather than separate copies for each user.

A.0 Appendix A. Customization Macros

This appendix is to be used in conjunction with Chapter 6, "Customizing VS FORTRAN Version 2" in topic 6.0.

Subtopics

A.1 Guidelines for Invoking Macros

1. Column 1 must be blank.

2. The macro name may appear anywhere before column 72, but must precede the options by at least one blank.

3. The options are separated by commas, with no intervening blanks, and may be continued on any number of lines as long as column 72 contains a nonblank character and the data on the following line begins in column 16.

4. A comma must follow the last option on a line when a continuation line follows.

5. You only need to code the options whose default values you wish to change.

A.2 VSF2COM: Changes Compile-Time Option Defaults

The VSF2COM macro allows you to change the IBM-supplied default values for most of the compile-time options. The default values you assign will be assumed if the user does not override them.

The compile-time options listed in this manual are designed to be used at installation. They have a different format than the corresponding compile-time options in the *VS FORTRAN Version 2 Programming Guide*.

¦ **Note:** There are no operands to set the default values for the compile-time options AUTODBL, CI, DC, DIRECTIVE, EC, PARALLEL, ¦ SC, and DYNAMIC; therefore these options *cannot* be changed at installation time. They can, however, be changed at compile time.

Each row in Figure 33 shows incompatible compile-time options. If *any* of these combinations of values are specified, the VSF2COM macro instruction will terminate without producing a new ILX0OPTS module.

Figure 33. Incompatible Compile-Time Options at Installation

Option	Conflicting Option
DBCS=DBCS	FIPS=F | S
DBCS=DBCS	LANGLVL=66
DBCS=DBCS	SAA=SAA
FIPS=F | S	FLAG=W | E | S
FIPS=F | S	LANGLVL=66
FIPS=F | S	SAA=SAA
FIPS=F | S	SORCIN=FREE
LANGLVL=66	SAA=SAA
LANGLVL=77	NAME=name
OBJPROG=NOOBJECT, PUNCH=NODECK	SYM=SYM
OBJPROG=NOOBJECT	TEST=TEST
OPTIMIZ=1 | 2 | 3	TEST=TEST
SAA=SAA	SORCIN=FREE
SORLIST=NOSOURCE	SRCFLG=SRCFLG
TEST=TEST	SYMDUMP=NOSDUMP

Syntax of VSF2COM Macro

```
VSF2COM
    SYSTEM=OS/VS
    [,ADJ=IL(DIM) | IL(NODIM)]
    [,CHARLEN=number | 500]
    [,DATE=MDY | YMD]
    [,DBCS=DBCS | NODBCS]
```

```
      [,DDIM=DDIM | NODDIM]
      [,EMODE=EMODE | NOEMODE]
      [,FIPS=S | F | NOFIPS]
      [,FLAG=I | W | E | S]
      [,HALT=I | W | E | S]
      [,ICA=NOICA | ICA [ (
        [,MXREF ( S | L ) | NOMXREF ]
        [,CLEN | NOCLEN ]
        [,CVAR | NOCVAR ]
        [,MSG ( { NEW | NONE | ALL } ) ]
        [,MSGON (number1,number2,...) |
         MSGOFF (number1,number2,...) ]
        [,RCHECK | NORCHECK ] ) ]
      [,IGNORE=DISABLED | ENABLED]
      [,INSTERR=NOLIST | LIST]
      [,LANGLVL=66 | 77]
      [,LINECNT=number | 60]
      [,NAME=name | MAIN]
      [,OBJATTR=RENT | NORENT]
      [,OBJID=GOSTMT | NOGOSTMT]
      [,OBJLIST=LIST | NOLIST]
      [,OBJPROG=OBJECT | NOOBJECT]
      [,OPTIMIZ=0 | 1 | 2 | 3 | NOOPTIMIZE]
      [,PTRSIZE=4 | 8 ]
      [,PUNCH=DECK | NODECK]
      [,SAA=SAA | NOSAA]
      [,SORCIN=FREE | FIXED]
      [,SORLIST=SOURCE | NOSOURCE]
      [,SORTERM=TERMINAL | NOTERMINAL]
      [,SORXREF=XREF | NOXREF]
      [,SRCFLG=SRCFLG | NOSRCFLG]
      [,STORMAP=MAP | NOMAP]
      [,SXM=SXM | NOSXM]
      [,SYM=SYM | NOSYM]
      [,SYMDUMP=SDUMP | NOSDUMP]
      [,TEST=TEST | NOTEST]
      [,TRMFLG=TRMFLG | NOTRMFLG]
      [,VECTOR=NOVECTOR | VECTOR [ (
       , [ REPORT [ ( optionlist ) ] | NOREPORT ]
       , [ INTRINSIC | NOINTRINSIC ]
       , [ IVA | NOIVA ]
       , [ REDUCTION | NOREDUCTION ]
       , [ SIZE ( { ANY | LOCAL | n } ) ]
       , [ MODEL ( { ANY | VF2 | LOCAL } ) ]
       , [ SPRECOPT | NOSPRECOPT ]
       , [ ANZCALL | NOANZCALL ]
       , [ CMPLXOPT | NOCMPLXOPT ] ) ]
```

The IBM-supplied default values are underlined in the following option list. If an option is not specified, its default value will be used.

SYSTEM = OS/VS
must **always** be specified.

ADJ = IL(DIM) | IL(NODIM)
Specifies whether the code for adjustable-dimensioned arrays is to be placed inline--IL(DIM), or computations are to be done via

library call--IL(NODIM). Inline placement may result in faster run time. The library call may result in slower run time, but it checks whether the lower bound is greater than the upper bound. If it is, message AFB187I will be issued. You may also specify IL(NODIM) as NOIL.

CHARLEN = number | 500

Specifies the maximum length for any character variable, character array element, or character function. Specify number as an integer between 1 and 32767. Within a program unit, you cannot specify a length for a character variable, array element, or function greater than the CHARLEN specified.

Note: Changing CHARLEN can change the size of your object file.

DATE = MDY | YMD

Specifies the format of the date to be printed by the compiler.

MDY

Specifies that DATE is to be in the format mmddyy (m=month, d=day, y=year).

YMD

Specifies that DATE is to be in the format yymmdd (y=year, m=month, d=day).

DBCS = DBCS | NODBCS

Specifies whether or not the source file contains double byte characters in symbolic names and in character constants. DBCS indicates that the source file may contain double byte characters; NODBCS indicates that it does not.

DDIM | NODDIM

Specifies whether pointee arrays that specify object-time dimensions are to be evaluated at each array element reference (DDIM) or at the time of entry into the subprogram (NODDIM).

For dynamically dimensioned arrays (DDIM), the array declarations can contain any integer variable of length 4 or 2. These variables are not restricted to appearing in a COMMON block or as dummy arguments, but the value of the integer variables must be known when the array element containing it is referenced.

Dynamically dimensioned arrays can be used in a Fortran main program.

EMODE = EMODE | NOEMODE

Specifies whether the code that is compiled for a subroutine of function can receive arguments that reside in an extended common block.

FIPS = S | F | NOFIPS

Specifies whether or not to flag standard language, and, if so, specifies the standard language flagging level:

S

Specifies subset standard language flagging.

F

Specifies full standard language flagging.

NOFIPS

Specifies no standard language flagging.

Items not defined in the current American National Standard are flagged. Flagging is valuable only if you want to write a program that conforms to the American National Standard for FORTRAN implemented in LANGLVL(77).

FLAG = I | W | E | S

Specifies the level of diagnostic messages to be written.

I

Specifies that all messages, including informational messages (return code 0 or higher), are to be written.

W

Specifies that warning messages (return code 4 or higher) are to be written.

E
Specifies that error messages (return code 8 or higher) are to be written.

S
Specifies that severe error messages (return code 12 or higher) are to be written.

FLAG allows you to suppress messages that are below the level desired. Thus, if you want to suppress all messages that are warning or informational, specify FLAG = E.

¦ HALT (I | W | E | S)
¦ Causes termination of the compile after any phase if the compiler return code is at or above the specified level.

¦ **I** Selects return code 0 (Informational diagnostic level).
¦ **W** Selects return code 4 (Warning diagnostic level).
¦ **E** Selects return code 8 (Error diagnostic level).
¦ **S** Selects return code 12 (Severe error diagnostic level).

¦ The HALT(I) option terminates the compile after the first compiler phase, which is the syntax analysis phase.

¦ If a compile of a source file is terminated because of the HALT option, an object file is not produced.

ICA=NOICA | ICA [(
[,MXREF (S | L) | NOMXREF]
[,CLEN| NOCLEN]
[,CVAR |NOCVAR]
[,MSG ({NEW | NONE | ALL })]
[,MSGON (number1,number2,...) |
MSGOFF (number1,number2,...)]
[,RCHECK | NORCHECK])]
Specifies whether intercompilation analysis (ICA) is to be performed, and controls the intercompilation analysis output. Specify ICA when you have a group of separately-compiled programs and subprograms that you want to process together and you need to know if there are any conflicting external references.

MXREF(S|L) | NOMXREF
Specifies whether to produce external cross-reference listings. To produce the shortest MSREF listing, use the default suboption with MXREF(S).

S
Eliminates names from the list of references that make no references and are not referenced by other subprograms.

L
Includes names in the list of references that make no references and are not referenced by other subprograms.

CLEN | NOCLEN
Specifies whether to check the length of named common blocks.

CVAR | NOCVAR
Specifies whether usage information for variables in a named common block is to be collected.

MSG({NEW | NONE | ALL})
Specifies the type of diagnostic messages to be printed.

NEW
Specifies that only messages about the new compilations will appear on the printout.

NONE
Specifies that only messages about deleting entries in an intercompilation analysis file will appear on the printout.

ALL
Specifies that all messages will be printed.

MSGON (number1,number2,...) | MSGOFF(number1,number2,...)
Specifies the intercompilation analysis messages to be issued. MSGON and MSGOFF are mutually exclusive. With MSGON, the messages you specify are issued; all others are suppressed. With MSGOFF, the messages you specify are suppressed; all others are issued. If you specify neither MSGON nor MSGOFF, all messages are issued.

number
The message number; for example, 61 for message ILX0061I.

RCHECK | NORCHECK
Specifies whether to produce a report of recursive subroutine calls or function invocations. If you specify RCHECK, and recursion is detected, ICA will print the sequence of the calls that might produce the recursion.

IGNORE = DISABLED | ENABLED
Specifies whether or not the IGNORE vector directive will be made available to the user. Users will have access to the IGNORE directive only if the IGNORE installation option is ENABLED. The IGNORE directive allows the application programmer to specify that certain vectorization dependencies do not exist. For further details, see *VS FORTRAN Version 2 Programming Guide*, SC26-4222. Note that there is no corresponding compile-time option.

INSTERR = NOLIST | LIST
Specifies whether or not to list the messages that could be issued by the VSF2COM macro. If LIST is chosen, all possible messages are listed and no object file is produced. When LIST is specified, a return code of 16 is generated by the assembler.

LANGLVL = 66 | 77
Specifies the language level at which the input source program is written.

66
Specifies the old Fortran level--the 1966 language standard plus IBM extensions.

77
Specifies the current Fortran level--the 1977 language standard plus IBM extensions.

LINECNT = number | 60
Specifies the maximum number of lines on each page of the printed source listing. Specify number as an integer between 7 and 32765. The advantage of using a large LINECNT number is that there are fewer page headings to look through if you are using a terminal. Your printed output will run together from page to page without a break.

NAME = name | MAIN
Can only be specified when LANGLVL(66) is chosen. It specifies the name that is generated on the output and the name of the CSECT generated in the object module. It applies only to main programs.

OBJATTR = RENT | NORENT
Specifies whether or not reentrant object code is to be generated by the compiler.

OBJID = GOSTMT | NOGOSTMT
Specifies whether or not internal statement numbers (for traceback purposes) are to be generated for a call sequence to a subprogram.

OBJLIST = LIST | NOLIST
Specifies whether or not the object module listing is to be produced.

OBJPROG = OBJECT | NOOBJECT
Specifies whether or not the object module is to be produced. If OBJECT is specified, it requires an object output file.

OPTIMIZ = 0 | 1 | 2 | 3 | NOOPTIMIZE
Specifies the optimizing level to be used during compilation.

0
Specifies no optimization.

1
Specifies register and branch optimization.

2
Specifies partial code-movement optimization, code movement that cannot introduce logic changes into the program.

3
Specifies full code-movement optimization, which can possibly introduce logic changes into the program.

NOOPTIMIZE
Specifies no optimization.

Note: If you are debugging your program, it is advisable to use NOOPTIMIZE. To create more efficient code and, therefore, a shorter run time, but usually a longer compile time, use OPTIMIZE(2) or (3).

¦ **PTRSIZE (4|8)**
¦ Sets the default length for pointer variables to 4 or 8 bytes. This can be overridden by explicit typing. Be aware that changing the
¦ pointer size will alter any common or equivalence association.

PUNCH = DECK | NODECK
Specifies whether or not the object module is to be produced in card-image format. If DECK is specified, it requires a punch output file.

SAA = SAA | NOSAA
Specifies whether or not the compiler is to flag language elements that are not part of the Systems Application Architecture (SAA) Fortran common programming interface. SAA causes flagging to be performed; NOSAA disables flagging.

SORCIN = FREE | FIXED
Specifies whether the input source program is to be in free format or in fixed format.

SORLIST = SOURCE | NOSOURCE
Specifies whether or not the source listing is to be produced.

SORTERM = TERMINAL | NOTERMINAL
Specifies whether or not error messages and compiler diagnostics are to be written on the terminal or a SYSTERM data set.

Note: If your users are compiling in a batch environment and are not using a SYSTERM data set, specify NOTERMINAL to avoid messages about having no terminal online.

SORXREF = XREF | NOXREF
Specifies whether or not a cross-reference listing of all variables and labels in the source program is to be produced.

SRCFLG = SRCFLG | NOSRCFLG
Allows error diagnostics to be inserted into the source listing immediately following the statement in error.

STORMAP = MAP | NOMAP
Specifies whether or not a table of source program names and statement labels is to be written.

SXM = SXM | NOSXM
Improves readability of XREF or map listing output at a terminal. SXM formats listing output for an 80-character wide terminal screen; NOSXM formats listing output for a printer.

SYM = SYM | NOSYM

Invokes the production of SYM cards in the object text file. The SYM cards contain location information for variables within a Fortran program.

SYMDUMP = SDUMP | NOSDUMP

Specifies whether or not symbol table information is to be generated in the object module and in the object module listing. If SYMDUMP=SDUMP is specified, the SDUMP suboption ISN will be in effect. This specifies that SDUMP tables be generated using internal statement numbers.

TEST = TEST | NOTEST

Specifies whether or not to create input for VS FORTRAN Version 2 Interactive Debug and symbol table information.

TRMFLG = TRMFLG | NOTRMFLG

Presents the statement in error and the diagnostic message together, whenever possible, on your terminal.

VECTOR [(
[REPORT [(optionlist)] | NOREPORT]
[INTRINSIC | NOINTRINSIC]
[IVA | NOIVA]
[REDUCTION | NOREDUCTION]
[SIZE ({ ANY | LOCAL | n })]
[MODEL ({ ANY | VF2 | LOCAL })]
[SPRECOPT | NOSPRECOPT]
[ANZCALL | NOANZCALL]
[CMPLXOPT | NOCMPLXOPT])]
| NOVECTOR

Specifies whether to invoke the vectorization process, which produces programs that can use the speed of the vector facility. VECTOR instructs the compiler to transform eligible statements in loops into vector instructions. As much of a loop as possible is translated into vector object code and the remainder is translated into scalar object code.

Vectorization requires that the optimization level in effect be OPT(2) or OPT(3). If the optimization level is not OPT(2) or OPT(3), the compiler upgrades the optimization level to OPT(3) for you. However, the use of DEBUG or invalid Fortran statements downgrades the optimization level. As a result, the optimization level may be downgraded to OPT(0) and no vectorization occurs.

The VECTOR compile-time option includes the following suboptions:

REPORT [(optionlist)] | NOREPORT

Specifies whether to produce a report of vectorization information. REPORT instructs the compiler to either display the report on a terminal screen, or provide the information in a printed report.

The format and content of the report varies, depending on the REPORT options specified. In general, the report consists of a listing of source statements. Additional information is supplied in the margins of the listing and in tables at the end of the listing; for example, brackets that indicate loop nesting, flags that identify vectorized and nonvectorized loops, and messages that give more detailed information about the vectorization process.

If both PARALLEL and VECTOR are specified and the REPORT suboption is specified in one or both of the options, a single listing containing the vector and parallel reports is produced.

optionlist

Where the options specified are one or more of the strings, TERM, LIST, XLIST, SLIST, or STAT, separated by blanks or commas.Figure 34 in topic A.2 shows the options, their abbreviations, and describes the output produced by each option.

Figure 34. VECTOR(REPORT) Options

Option	Abbreviation	Description
TERM	TE	Produces a vector report at the terminal when TRMFLG is specified.

LIST	LI	Produces a simple format of the vector report on an output listing. This suboption will help you understand how the statements and loops in the pseudo-assembler listing have been reordered.
XLIST	XL	Produces an extended format of the vector report on an output listing. Use XLIST when you want to tune your program to increase vectorization. It will help you understand why statements did or did not vectorize.
SLIST	SL	Produces a vector report, in a format similar to that produced by the SOURCE compile-time option, on an output listing. Use SLIST in conjunction with the VEC(IVA) option when you are trying to tune your program to improve overall performance. This allows you to use the vector tuning tools available in the IAD. Use SLIST in conjunction with the NOSOURCE option when you want to create a single listing for archive purposes.
STAT	ST	Produces a vector statistics table on an output listing. Use STAT when you are tuning a program for performance without the IAD; it will tell you about some of the assumptions made by the compiler that could have affected its economic decisions.

You can specify the options in any order; however, in the printed listing, the sections of the report appear in the following order: LIST, XLIST, SLIST, STAT.

The above reports can be requested individually or in any combination.

INTRINSIC | NOINTRINSIC

Specifies whether out-of-line intrinsic function references are to be vectorized.

The VS FORTRAN Version 2 math library routines (VSF2FORT) have been revised to be more accurate. The results generated by these new routines may be different from the results generated by the old VS FORTRAN Version 1 standard math routines (VSF2MATH).

For scalar out-of-line intrinsic function references in your program, you can choose which math library to use by accessing libraries in the desired order. If the intrinsic function references in your program are vectorized, however, the new VS FORTRAN Version 2 math library routines will always be used.

Therefore, if you wish to always use the old math routines for compatibility of results, you should not allow out-of-line intrinsic function references to vectorize.

If you use the INTRINSIC suboption, out-of-line intrinsic function references in your program are eligible for vectorization. If the new math library routines are used, the scalar results will be the same as the vector results. The new math routines yield the

same results in both vector and scalar.

If you use the NOINTRINSIC suboption, out-of-line intrinsic function references in your program are vectorized. The scalar math routines of the library you have accessed first will be used.

IVA | NOIVA
Specifies whether to produce a program information file. This file is required by Interactive Debug if you use the interactive vectorization aid functions.

REDUCTION | NOREDUCTION
Specifies whether reduction functions are to be vectorized.

The translation of vector operations from scalar to vector code may produce different results. For example, because floating-point addition is not associative, vector code operations may not be performed in scalar processing order. This causes the results to be dependent on the order of accumulation of the floating-point data.

For more information, see *IBM Enterprise Systems Architecture/370 and System/370 Vector Operations*.

SIZE ({ANY | LOCAL | n})
Specifies the section size to be used to perform vector operations.

Specifying SIZE(ANY) causes the compiler to generate object code to use the section size of the computer on which the routine is running. The code may be slightly less efficient as that generated by SIZE(LOCAL) or SIZE(**n**) but you can move the program to a computer with a different section size without recompiling.

When you specify SIZE(LOCAL) the compiler uses the specified section size of the computer that compiled the program. If the section size is 0 or -1, the compiler uses the IBM-supplied default, ANY.

SIZE(**n**) allows you to specify an explicit section size. Using LOCAL or **n** produces slightly more efficient object code. However, if the specified section size is different from that of the computer's actual section size the program terminates upon the first processing of the routine compiled with the incompatible section size. The library issues a diagnostic message with a return code of 16.

If the specified section size is invalid--not a power of 2, or less than 8--the compiler issues an informational message and uses the IBM-supplied default, ANY.

MODEL ({ANY | VF2 | LOCAL })
Identifies the level of the ES/9000 vector facility on which the compiled program will be executed. The compiler generates code to use the enhanced features of that level.

When a program has been compiled for a specific level of the vector facility, the VS FORTRAN run-time vector support will allow the program to be executed only on a processor that can support that level of the vector facility.

ANY
Specifies that any level of the vector facility can be used to execute this program. The program will run on any IBM ES/3090 or ES/9000 processor that has the vector facility installed.

VF2
Specifies that the program will be executed on an ES/9000 processor with the basic level of the enhanced vector facility. Code compiled with this option can execute faster than code generated with MODEL(ANY).

LOCAL
Specifies that the program will be compiled for the level of the vector facility installed on the processor where it is compiled.

If the compiling processor

- Does not have the vector facility installed,
- Is an ES/3090 processor
- Is an ES/9000 processor without an enhanced vector facility

MODEL(LOCAL) is equivalent to MODEL(ANY).

If the compiling processor is an ES/9000 with the basic level of the enhanced vector facility, MODEL(LOCAL) is equivalent to MODEL(VF2).

SPRECOPT | NOSPRECOPT
Allows single-precision multiply-and-add and multiply-and-subtract sequences to be generated. If SPRECOPT is specified, the vectorization process will look for single-precision multiplications followed by single-precision additions or subtractions, and attempt to perform these sequences with a vector multiply-and-add or multiply-and-subtract instruction. These calculations give double-precision results, and use double-precision values for the intermediate product. Processing is faster and the result is more accurate than for two separate single-precision operations, but you should be aware that the result might be different from what you expected.

ANZCALL | NOANZCALL
Specifies whether an analysis of CALL statements and user function references within DO loops is to be performed when the VECTOR option is in effect. While greater vectorization might be achieved with ANZCALL, compilation time will increase for eligible loops.

CMPLXOPT | NOCMPLXOPT
Specifies whether vectorization is to optimize loading and storing of single-precision complex (COMPLEX*8) data using long-precision real vector instructions.

A.3 VSF2UAT: Changes I/O Unit Number Option Defaults

The VSF2UAT macro allows you to specify default values for information that is required by the run-time input/output routines of the VS FORTRAN Version 2 Library.

Syntax of VSF2UAT Macro: Statement Form

```
AFBVUAT VSF2UAT
     [DECIMAL=PERIOD | COMMA]
     [,PUNCH=number | 7 ]
     [,ERRMSG=number | 6 ]
     [,PRINTER=number | 6 ]
     [,READER=number | 5 ]
     [,UNTABLE=number | 99]
     [,DEVICE=device-name | SYSDA]
```

You may code as many VSF2UAT statements as you need. However, you must use the following form of VSF2UAT as the final macro instruction when you change the I/O unit options and default values. See "Examples of Changing Default Values" in topic 6.6.2.

Syntax of VSF2UAT Macro: Final Statement

```
         VSF2UAT   TYPE=FINAL
```

The IBM-supplied default values are underlined in the following option list. If an option is not specified, its default value will be used.

DECIMAL = PERIOD | COMMA
Specifies the character to be used as the decimal indicator in printed output.

PUNCH = number | 7
Specifies, for LANGLVL(66) only, the standard I/O unit number for the PUNCH statement to send data to the card punch. The number specified must be between 0 and 99 or the value specified for the UNTABLE option, for UNTABLE values less than or equal to 99. It must not be the same as the number specified for ERRMSG, PRINTER, or READER.

ERRMSG = number | 6
Specifies the standard I/O unit number for the error messages generated by VS FORTRAN Version 2. The number specified must be between 0 and 99 or the value specified for the UNTABLE option, for UNTABLE values less than or equal to 99. It must not be the same as the number specified for PUNCH or READER.

PRINTER = number | 6
Specifies the standard I/O unit number for the print statement, and with any WRITE statement specifying an installation-dependent form of the unit. The number specified must be between 0 and 99 or the value specified for the UNTABLE option, for UNTABLE values less than or equal to 99. It must not be the same as that specified for PUNCH and READER. It can be the same number specified for ERRMSG.

READER = number | 5
Specifies the standard I/O unit number for any READ statement specifying an installation-dependent form of the unit. The number specified must be between 0 and 99 or the value specified for the UNTABLE option, for UNTABLE values less than or equal to 99. It must not be the same as the number specified for either PUNCH, ERRMSG, or PRINTER.

UNTABLE = number | 99
Specifies the largest unit number you can include in a VS FORTRAN program. It can be specified as any integer between 8 and 2000.

DEVICE = device-name | SYSDA

Specifies a unit address, group name, or device type for a DASD device. The unit address is 3 or 4 hexadecimal characters, consisting of the channel, control unit, and unit number. The group name may be any name that is defined during system generation for a DASD device, such as SYSDA, DISK or an IBM-supplied name, such as 3330, 3330-1, 3350, or 3380.

The default **SYSDA** applies to all units except PUNCH, ERRMSG, PRINTER, and READER units. **Device-name** is used to allocate a new data set or an existing one which is not in the catalogue.

The programmer can override the **device-name** without having to customize the Unit Attribute Table by specifying a new device in the CALL FILEINF service routine.

TYPE = FINAL

Is the required last statement of the VSF2UAT macro.

Note: VS FORTRAN Version 2 refers to PUNCH, ERRMSG, PRINTER and READER as the standard I/O units.

A.4 VSF2UNIT: Identifies I/O Units to have DCB Defaults

The VSF2UNIT macro allows you to specify a single unit, or group of units, that are to be assigned DCB default values. It is used in conjunction with the VSF2DCB macro.

```
+--- Syntax of VSF2UNIT Macro ------------------------------------------+
¦                                                                       ¦
¦ VSF2UNIT                                                              ¦
¦     unitno | ( unitno [,qty] )                                        ¦
¦     ,DCBSET = label                                                   ¦
¦                                                                       ¦
¦                                                                       ¦
+-----------------------------------------------------------------------+
```

unitno
Specifies the unit number, or the first in a series of consecutive unit numbers, that are to have DCB default values assigned.

qty
Specifies, if there is more than one, the number of consecutive unit numbers, beginning with **unitno**, that are to have DCB default values assigned.

DCBSET=label
Specifies the label that is applied to the units designated by **unitno**|(**unitno**[,**qty**]). The VSF2DCB macro must have a corresponding label.

A.5 VSF2DCB: Identifies DCB Default Information

The VSF2DCB macro allows you to specify DCB default information for the I/O units identified by the DCBSET=label option of the VSF2UNIT macro.

Syntax of VSF2DCB Macro

```
[label]  VSF2DCB
     [,SFBUFNO=number | 2]
     [,SUBUFNO=number | 2]
     [,SFBLKSI=number | 800]
     [,SUBLKSI=number | 800]
     [,SFLRECL=number | 800]
     [,SULRECL=number. | -1]
     [,SFRECFM=char | U]
     [,SURECFM=char | VS]
     [,SFMAXRE=number | 100]
     [,SUMAXRE=number | 100 ]
     [,DMAXRE=number | 100]
```

label
Specified in the VSF2UNIT macro to identify the I/O units that are to be assigned DCB default values.

If **label** is omitted, the DCB data is assigned to all units defined in the default table by the VSF2UAT macro, but which have not been defined by the VSF2UNIT macro. If any of the units defined in the attribute table do not have their own associated DCBSET coded, you must provide a VSF2DCB macro **without a label** to apply defaults to these units.

SFBUFNO=number | 2
Specifies the default value for the number of buffers for sequential formatted files on DASD or tape. Number must be a value greater than or equal to 1 and less than or equal to 255.

SUBUFNO=number | 2
Specifies the default value for the number of buffers for sequential unformatted files on DASD or tape. Number must be a value greater than or equal to 1 and less than or equal to 255.

SFBLKSI = number | 800
Specifies the block size for sequential formatted files. **Number** is an integer expression of length 4 bytes; valid range of the blocksize is from 1 to 32760.

SUBLKSI = number | 800
Specifies the block size for sequential unformatted files. **Number** is an integer expression of length 4 bytes; valid range of the blocksize is from 1 to 32760.

SFLRECL = number | 800
Specifies the logical record length for sequential formatted files. **Number** is an integer expression of length 4 bytes; valid range is from 1 to 32756 for variable record formats (SURECFM= V, VA, VB, or VBA), or 1 to 32760 for all other record formats.

SULRECL = number | -1
Specifies the logical record length for sequential unformatted files. **Number** is an integer expression of length 4 bytes; valid range is from 1 to 32756 for variable record formats (SURECFM= V, VA, VB, VBA, VS, or VBS), or 1 to 32760 for all other record formats or -1, which specifies an unlimited record length. -1 is valid for SURECFM=VS or VBS formats.

SFRECFM = char | U
Specifies the record format for sequential formatted files. The value of **char** must be F, FA, FB, FBA, V, VA, VB, VBA, U, or UA. For more information on I/O, see the *VS FORTRAN Version 2 Programming Guide*.

SURECFM = char | VS

Specifies the record format for sequential unformatted files. The value of **char** must be F, FA, FB, FBA, V, VA, VB, VBA, VS, VBS, U, or UA. For more information on I/O, see the *VS FORTRAN Version 2 Programming Guide*.

SFMAXRE = number | 100

Specifies the amount of space to be converted into blocks in a sequential formatted file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. **Number** is an integer expression of length 4. See the *VS FORTRAN Version 2 Programming Guide* under MAXREC for information on how space is converted to blocks.

SUMAXRE = number | 100

Specifies the amount of space to be converted into blocks in a sequential unformatted file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. **Number** is an integer expression of length 4. See the *VS FORTRAN Version 2 Programming Guide* under MAXREC for information on how space is converted to blocks.

DMAXRE = number | 100

Specifies the amount of space to be converted into blocks in a direct file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. **Number** is an integer expression of length 4. See the *VS FORTRAN Version 2 Programming Guide* for information on how space is converted to blocks.

A.6 VSF2PARM: Changes Run-Time Option Defaults

The VSF2PARM macro allows you to change the IBM-supplied default values for run-time options. The default values you assign will be assumed if the user does not override them.

¦ There are no operands to set the default values for the run-time options AUTOTASK, PARALLEL, and PARTRACE; therefore these ¦ options cannot be changed during installation. They can, however, be changed at run time.

¦ There are no operands in the VSF2PARM macro to set the default values for the run-time options ERRUNIT, RDRUNIT, PRTUNIT, and ¦ PUNUNIT. The default I/O unit values for these units can be changed during installation through the Unit Attribute Table. See ¦ "Changing the Unit Attribute Table Defaults" in topic 6.6.2.

Syntax of VSF2PARM Macro

```
VSF2PARM
     SCOPE = GLOBAL
     [,ABSDUMP | NOABSDUMP]
     [,CNVIOERR | NOCNVIOERR]
     [,DEBUG | NODEBUG]
     [,DEBUNIT(s1[,s2,...]) | NODEBUNIT]
     [,ECPACK | NOECPACK]
     [,FAIL(ABEND | RC |ABENDRC)]
     [,FILEHIST | NOFILEHIST]
     [,INQPCOPN | NOINQPCOPN]
     [,IOINIT | NOIOINIT]
     [,OCSTATUS | NOOCSTATUS]
     [,RECPAD[(ALL)] | NORECPAD]
     [,SPIE | NOSPIE]
     [,STAE | NOSTAE]
     [,XUFLOW | NOXUFLOW]
```

The IBM-supplied default values are underlined in the following option list. If an option is not specified, its default will be used, with the exception of the SCOPE option, which must always be specified.

SCOPE = GLOBAL

Required to replace the global run-time options table AFBVGPRM, which supplies default values for all users of the VS FORTRAN Version 2 Library.

There is no default value for this option. Thus SCOPE=GLOBAL must always be specified.

ABSDUMP | NOABSDUMP

Specifies whether or not the post-abend symbolic dump information is printed.

ABSDUMP

Causes the post-abend symbolic dump information to be printed in the event of an abnormal termination.

NOABSDUMP

Suppresses the printing of the post-abend symbolic dump information.

CNVIOERR | NOCNVIOERR

Specifies whether input conversion errors will be treated as I/O errors.

CNVIOERR

Causes ERR and IOSTAT to recognize conversion errors as I/O errors.

NOCNVIOERR
Causes conversion errors not to be treated as I/O errors. ERR and IOSTAT have no effect for these errors.

DEBUG | NODEBUG
Specifies whether or not interactive debug will be invoked.

DEBUG
Causes interactive debug to be invoked.

NODEBUG
Specifies that interactive debug will not be invoked.

DEBUNIT | NODEBUNIT
Specifies whether or not Fortran unit numbers will be treated as if connected to a terminal device.

DEBUNIT
Passes a list of Fortran unit numbers to the running environment. The specified unit numbers will be treated as if they were connected to a terminal device so that the interactive debug command TERMIO can be used to control whether the I/O is done by interactive debug or the library. The format of the option is

```
DEBUNIT(s1[,s2,...])
```

where **s** is a single unit number or a range of unit numbers. A range of unit numbers is expressed as **s1-s2**, where both **s1** and **s2** are unit numbers, and the ending unit number, **s2**, is not less than the starting number, **s1**.

The unit numbers specified must be one- to four-digit numbers within the range of numbers allowed on your system, as specified by the UNTABLE option of the VSF2UAT macro. See page A.3 for information on the UNTABLE option.

NODEBUNIT
Suppresses treating Fortran unit numbers as if connected to a terminal device.

ECPACK | NOECPACK
Specifies whether a data space should be filled with as many extended common blocks as possible before a new data space is allocated.

ECPACK
Specifies extended common blocks be placed into the fewest possible number of data spaces. This option reduces some of the overhead associated with referencing data spaces.

NOECPACK
Specifies that each extended common block be placed into a separate data space. As a result, reference errors made beyond the bounds of an extended common block might be more easily detected.

FAIL (ABEND | RC | ABENDRC)
Indicates how unsuccessfully executed programs are to be terminated: either by a nonzero return or by an abnormal termination (ABEND). The suboption of the FAIL option may have the following meanings.

ABEND
Causes the program to end by an abnormal termination (ABEND) with a user completion code of 240.

RC

Causes the program to end normally but with a nonzero return code (16).

ABENDRC

Causes the program to end by abnormal termination (ABEND) when failure is because of a condition for which the operating system would usually cause an ABEND; and to end with a nonzero return code when failure is by some condition detected by VS FORTRAN.

FILEHIST | NOFILEHIST

Specifies whether to allow the file definition of a file referred to by a ddname to be changed at run time.

FILEHIST

Causes the history of a file to be used in determining its existence. In particular it checks to see whether:

The file was ever internally opened (in which case it exists)
The file was deleted by a CLOSE statement (in which case it does not exist).

When FILEHIST is specified, you cannot change the file definition of a file at run time and have the same results produced as previous VS FORTRAN releases.

NOFILEHIST

Causes the history of a file to be disregarded in determining its existence.

If you specify NOFILEHIST you should consider:

If you change file definitions at run time: the file is treated as if it was being opened for the first time. Note that before the file definition can be changed, the existing file must be closed.
If you do not change file definitions at run time: you must use STATUS=NEW to re-open an empty file that has been closed with STATUS=KEEP, because the file does not appear to exist to Fortran.

INQPCOPN | NOINQPCOPN

Specifies whether or not a unit is connected to a file when executing an INQUIRE by unit.

INQPCOPN

Specifies that, if a unit is connected to a file, even if it was preconnected and no I/O statement has been executed, a value of true is returned in the variable or an array element given in the OPENED specifier from an INQUIRE by unit statement.

NOINQPCOPN

Indicates that, if and only if an unit is internally open, a value of true is returned in the variable or an array element given in the OPENED specifier for an INQUIRE by unit statement.

"Internally open" means that the unit is connected to a file by an OPEN statement, or if the unit has been preconnected, that a READ, WRITE, PRINT, REWIND, or ENDFILE statement has been successfully executed.

IOINIT | NOIOINIT

Specifies whether or not the normal initialization for I/O processing will occur during initialization of the run-time environment.

IOINIT

Causes the normal initialization for I/O processing to occur during initialization of the run-time environment.

NOIOINIT

Suppresses initialization for I/O processing. This means that the error message unit will not be opened during initialization of the run-time environment. However, this does not prevent I/O from occurring on this or on any other unit. (Such I/O may fail if proper DD statements are not given.)

OCSTATUS | NOOCSTATUS

Specifies whether file existence will be checked during the running of OPEN statements, whether files are deleted from their storage media, and whether files that have been closed can be reconnected without an OPEN statement.

OCSTATUS

Specifies:

1. File existence will be checked for consistency with the OPEN statement specifiers STATUS='OLD' and STATUS='NEW'.

2. File deletion will occur when the CLOSE statement specifier STATUS='DELETE' is given (on devices which allow deletion).

3. A preconnected file will be disconnected when a CLOSE statement is given or when another file is opened on the same unit. It can be reconnected only by an OPEN statement when there is no other file currently connected to that unit.

NOOCSTATUS

Specifies:

1. File existence will not be checked for consistency with the OPEN statement specifiers STATUS='OLD' and STATUS='NEW'.

2. File deletion will not occur when the CLOSE statement specifier STATUS='DELETE' is given.

3. A preconnected file will be disconnected when a CLOSE statement is given or when another file is opened on the same unit. It can be reconnected by a sequential READ or WRITE, BACKSPACE, OPEN, REWIND, or ENDFILE statement when there is no other file currently connected to that unit.

RECPAD[(ALL)] | NORECPAD

Specifies whether a formatted input record is padded with blanks.

RECPAD

Causes a formatted input record within an internal file or a varying/undefined length record (RECFM=U or V) external file to be padded with blanks when an input list and format specification require more data from the record than the record contains. Blanks added for padding are interpreted as though the input record actually contains blanks in those fields. If ALL is specified, a formatted input record is padded regardless of the record format of the file.

NORECPAD

Specifies that an input list and format specification must not require more data from an input record than the record contains. If more data is required, error message AFB212I is issued.

SPIE | NOSPIE

Specifies whether or not the run-time environment will take control when a program interrupt occurs.

SPIE

Specifies that the run-time environment takes control when a program interrupt occurs.

NOSPIE

Specifies that the run-time environment does not take control when a program interrupt occurs.

Recommendation: If you are using the DEBUG option, specify SPIE to avoid termination of interactive debug when an interrupt occurs.

If you specify NOSPIE, various run-time functions that depend on a return of control after a program interrupt are not available. These include the following:

The messages and corrective action for a floating-point overflow

The messages and corrective action for a floating-point underflow interrupt (unless the underflow is to be handled by the hardware based upon the XUFLOW option)

The messages and corrective action for a floating-point or fixed-point divide exception

The simulation of extended precision floating-point operations on processors that do not have these instructions

The realignment of vector operands that are not on the required storage boundaries and the re-running of the failed instruction.

```
Instead of the corrective action, abnormal termination results.
In this case, the STAE or NOSTAE option that is in effect governs
whether or not the VS FORTRAN run-time environment gains control
at the time of the abend.
```

STAE | NOSTAE

Specifies whether or not the run-time environment will take control if an abnormal termination occurs.

STAE

Specifies that the run-time environment will take control when an abnormal termination occurs.

NOSTAE

Specifies that the run-time environment does not take control when an abnormal termination occurs.

Recommendation: If you are using the DEBUG option, specify STAE to avoid termination of interactive debug when an interrupt occurs.

If NOSTAE is specified, abnormal termination is handled by the operating system rather than by the VS FORTRAN run-time environment. In this case the following occurs:

Message AFB240I, which shows the PSW and register contents at the time of the abend, is not printed. However, this information will be provided by the operating system.

The indication of which Fortran statement caused the failure will not be printed.

The traceback of the routines will not be printed.

The post-abend symbolic dump will not be printed even with the option ABSDUMP in effect.

Certain exceptional conditions handled by the run-time environment or by the debugging device cause system abends rather than VS FORTRAN messages. For example, some errors that occur during running of an OPEN statement result in a system abend rather than the printing of message AFB219I, which allows the program to possibly continue running.

If the QUIT command is used in an interactive debug attention exit to terminate a program during a TSO debugging session, a user ABEND 500 occurs instead of the normal termination of the run-time environment.

An MTF subtask that terminates unexpectedly causes a user ABEND 922 in the main task rather than message AFB922I.

XUFLOW | NOXUFLOW

Specifies whether or not an exponent underflow will cause a program interrupt.

XUFLOW

Allows an exponent underflow to cause a program interrupt, followed by a message from the VS FORTRAN Version 2 Library, followed by a standard fixup.

NOXUFLOW

Suppresses the program interrupt caused by an exponent underflow. The hardware sets the result to zero.

A.7 VSF2UOPT: Customizes the Error Option Table

The VSF2UOPT macro allows you to customize the error option table as follows:

Adding new error messages to the table, without changing existing ones, by coding the **VSF2UOPT Required Macro Instruction,** followed by an END statement.

Changing existing error messages in the table, with or without adding new ones, by coding the **VSF2UOPT Required Macro Instruction,** followed by the necessary number of optional macro instructions, followed by an END statement.

For information on IBM-supplied error messages, refer to "Extended Error-Handling Subroutines and Error Option Table" in *VS FORTRAN Version 2 Language and Library Reference*.

Syntax of VSF2UOPT Required Macro Instruction

```
VSF2UOPT
    [ADDNTRY = n]
```

ADDNTRY=n
Is a positive integer specifying the number of new error message numbers to be added to the error option table. Additional error message numbers will begin at 500 and continue sequentially, up to a maximum of 899. If you want to change existing messages but do not want to add new ones, omit ADDNTRY=**n**.

n
is a positive integer between 1 and 598.

Syntax of VSF2UOPT Optional Macro Instruction

```
VSF2UOPT
    MSGNO = (ermsno[,qty])
    [,ALLOW = errs]
    [,INFOMSG = YES | NO]
    [,IOERR = YES | NO]
    [,MODENT = YES | NO]
    [,PRINT = prmsg]
    [,PRTBUF = YES | NO]
    [,TRACBAK = YES | NO]
    [,USREXIT = exitname]
```

The MSGNO option must always be specified. The default values of the five options INFOMSG, IOERR, MODENT, PRTBUF, and TRACBAK vary according to the following conditions:

If the value of MSGNO specifies an IBM-supplied message number, and *none* of the five options is changed, then the default values are found in "Extended Error-Handling Subroutines and Error Option Table" of *VS FORTRAN Version 2 Language and Library Reference*.

If eithe

- the value of MSGNO specifies an IBM-supplied message number, and *one or more* of the five options is changed, or

- the value of MSGNO specifies a new message number,

then the default values for the *unspecified* options are the following:

INFOMSG
NO
IOERR
NO
MODENT
YES
PRTBUF
NO
TRACBAK
YES

MSGNO = (ermsno[,qty])
Specifies which error messages are affected by the default changes.

ermsno
Specifies either one message number, or the first error message number in a series of consecutive numbers.

qty
Specifies, if there is more than one, the number of consecutive error message numbers, beginning with **ermsno**.

For example, if the option is coded **MSGNO=(153)**, then the default values for message 153 will be changed. If the option is coded **MSGNO=(153,4)**, then the default values for messages 153 through 156 will be changed.

ALLOW = errs
Specifies the number of times the error may occur before the program is terminated.

errs
Specifies the number of errors allowed. To specify an exact number of errors allowed, errs must be a positive integer with a maximum of 255. A zero, or any number greater than 255, means the error can occur an unlimited number of times.

Note: Be aware that altering an error option table entry to allow "unlimited" error occurrence may cause a program to loop indefinitely.

If the value of MSGNO specifies an IBM-supplied message number, the default value for this option is listed in "Extended Error-Handling Subroutines and Error Option Table" of *VS FORTRAN Version 2 Language and Library Reference*. If the value of MSGNO specifies a new message number, the default value is 10.

INFOMSG = YES | NO
Specifies whether the message is an informational or an error message.

YES
Specifies that the message is informational only. In this case the following occurs:

No user error exit is taken.

The value of ALLOW is ignored. Running will not terminate, even if it reaches the designated number of errors allowed.

The error summary printed after termination of your program does not include a count of the number of times the condition occurred.

NO
Specifies that the message is an error message.

IOERR = YES | NO
Specifies whether or not this error message represents an I/O error for which error counting is to be suppressed when an ERR or IOSTAT option is given on the I/O statement.

YES
Specifies that if an ERR or IOSTAT option is given, the occurrence of the error is not to be counted toward the maximum number specified by the ALLOW option above. This should be specified only for those errors, listed in *VS FORTRAN Version 2 Language and Library Reference*, for which the ERR and IOSTAT options are honored.

NO
Specifies that the error occurrence is to be counted toward the maximum number of errors allowed.

MODENT = YES | NO
Specifies whether or not the ERRSET subroutine may be used to modify the error option table entry for this message.

YES
Specifies that the entry may be modified.

NO
Specifies that the entry may not be modified.

If you code a YES value for an IBM-supplied error message whose default is NO, and you subsequently modify this entry using the ERRSET subroutine, you may receive undesirable results. Check the chapter "Extended Error-Handling Subroutines and Error Option Table" of *VS FORTRAN Version 2 Language and Library Reference*, to find out which message numbers have a "Modifiable Entry" value of NO.

PRINT = prmsg
Specifies the number of times the error message is to be printed. Subsequent occurrences of the error do not cause the message to be printed again.

prmsg
Specifies the number of times the message is to be printed. To specify an exact number of times printed, **prmsg** must be a positive integer, with a maximum of 254. A "0" means the message will not be printed. Specifying 255 means the message can be printed an unlimited number of times.

If the value of MSGNO specifies an IBM-supplied message number, the default value for this option is listed in the chapter "Extended Error-Handling Subroutines and Error Option Table" in *VS FORTRAN Version 2 Language and Library Reference*. If the value of MSGNO specifies a new message number, the default value is 5.

PRTBUF = YES | NO
Specifies whether or not the I/O buffer is to be printed following certain I/O errors.

YES
Specifies that the contents of the buffer are to be printed.

NO
Specifies that the contents of the buffer are not to be printed.

This option applies only to IBM-supplied error messages. Do not code YES unless the IBM-supplied default for this error message number already allows the buffer to be printed. Check the chapter "Extended Error-Handling Subroutines and Error Option Table" in *VS FORTRAN Version 2 Language and Library Reference* to find out which message numbers have a "Print Buffer" value of YES.

TRACBAK = YES | NO

Specifies whether or not a module traceback listing is to be printed following the error message.

YES

Specifies that the traceback listing is to be printed.

NO

Specifies that the traceback listing is not to be printed.

USREXIT = exitname

Specifies the user error exit routine that is invoked following the printing of the error message.

exitname

Specifies the entry point name of the user error exit routine. The routine should not be written in VS FORTRAN and should be reentrant.

If the routine is specified here, instead of being specified as an option passed to the ERRSET subroutine, the routine is invoked when the error occurs for any user. In this case, the routine will be invoked, regardless of whether the ERRSET routine was used or not. (However, unless a MODENT value of NO is in effect, programs can still call ERRSET dynamically to specify their own exit routine instead of the one specified by USREXIT.)

For programs operating in link mode, the user error exit routine must be link-edited with all users' programs.

To make the user error exit routine available to users who operate in load mode, the routine must be included in the composite modules AFBVRENA and AFBVRENC. Then, if the user error exit routine must communicate with the VS FORTRAN Version 2 program in which the error was detected, it must do so using a dynamic common area, not a static one.

A.8 VSF2AMTB: Customizes an Auxiliary Error Option Table

The VSF2AMTB macro allows you to customize an auxiliary error option table. This table can be used by a product other than VS FORTRAN Version 2 that needs to link with it and that could make use of its error handling facility. Note that this auxiliary error option table is different from the regular error option table used only by VS FORTRAN Version 2.

To use the VSF2AMTB macro:

1. Define the name and scope of the table by coding the VSF2AMTB required macro instruction. The syntax is shown below.

2. Define table contents for individual error message entries. Follow the syntax for the VSF2AMTB required macro instruction with the necessary number of optional macro instructions, one for each message you wish to create, followed by an END statement. The syntax for the VSF2AMTB optional macro instruction is shown on page A.8.

3. After you invoke the necessary macros, assemble them. This will produce your auxiliary error option table, with a name of **pid**UOPT, where **pid** is the auxiliary product identifier you supplied on the VSF2AMTB required macro instruction shown below.

Refer to your auxiliary product's documentation to determine where and how to store your assembled table to make it part of the auxiliary product.

Syntax of VSF2AMTB: Required Macro Instruction

```
VSF2AMTB
     COMPID = pid,
     MSGNUM1 = firstnum,
     MSGNUM2 = lastnum
```

COMPID=pid
Specifies the first three characters of the table name. The macro concatenates these three characters with the characters UOPT, creating a name for the table of the form **pid**UOPT. The first character of **pid** must be a letter, the following two characters must be alphanumeric.

MSGNUM1=firstnum
Specifies the starting number of the table. The minimum value is 10000.

MSGNUM2=lastnum
Specifies the ending number of the table. The maximum value is 19999.

Note: There are no default values for any of the three options; they must always be specified.

Syntax of VSF2AMTB: Optional Macro Instruction

```
VSF2AMTB
     MSGNO = (ermsno,qty)
     [,ALLOW = errs]
     [,INFOMSG = YES | NO]
     [,MODENT = YES | NO]
     [,PRINT = prmsg]
     [,PRTBUF = YES | NO]
     [,TRACBAK = YES | NO]
     [,USREXIT = exitname]
```

+--+

The option list is identical to the option list of VSF2UOPT, shown on page A.7, except that VSF2AMTB does *not* have an IOERR option.

If you omit either of the following:

A macro instruction for any error message whose number is in the auxiliary table range you specified in the first macro instruction, or

Any individual option in a particular macro instruction

then the default values are the following:

```
ALLOW     10
INFOMSG   NO
MODENT    YES
PRINT     5
PRTBUF    NO
TRACBAK   YES
USREXIT   NO USER EXIT TAKEN
```

B.0 Appendix B. Composite Modules

This appendix is to be used in conjunction with "Rebuilding Composite Modules" in topic 6.4.

Subtopics

B.1 Reasons for Using Composite Modules

If programmers choose run-time loading of library modules (load mode), each module is loaded the first time it is used, unless it has been previously loaded. Because run-time performance suffers if a large number of library modules are individually loaded, they are combined into composite modules.

As part of its initialization procedure in load mode, the run-time library loads the appropriate composite modules listed in Figure 35 in topic B.2. The only modules that need to be loaded separately after initialization are those that are not contained in the composite modules.

At any time after installation, you may add or delete optional library modules from the composite modules to further tune your system. For example, if keyed access and direct access are not normally used at your site, you may choose not to place the modules that perform these functions in the composite load modules. This reduces the size of the composite modules. The run-time library will then have to load the direct access and keyed access I/O modules individually when they are needed. (These modules may reside in the link pack area so they don't need to be brought into your region, or they may be brought into your region from the library containing them.)

B.2 Characteristics of the Composite Modules

Figure 35. Composite Module Characteristics

Composite Module Name	Can Be Installed In an LPA	Used for MVS/SP Version 1	Used for MVS/XA or MVS/ESA	Location In Relation to 16M line (XA only)	Contents
AFBVRENA	Yes	No	Yes	Above	Reentrant library modules. Many library modules that previously resided in the VS FORTRAN Version 1 module IFYVRENT (below the 16M line) may now be placed in AFBVRENA for virtual storage constraint relief.
AFBVRENB	Yes	No	Yes	Below	Reentrant library modules
AFBVRENC	Yes	Yes	No	Not applicable	Reentrant library modules
AFBVRENP	Yes	Yes	Yes	Above	Reentrant parallel support library modules

B.3 Customization Tips

If either AFBVRENA and AFBVRENB, or AFBVRENC is not in the LPA (link pack area), it must be loaded into your region. Including all possible reentrant modules in them may require the region size to be larger than would otherwise be necessary.

If either AFBVRENA and AFBVRENB, or AFBVRENC is in the LPA, including a large number of the reentrant modules in these composite modules does not increase the user region size. However, the larger AFBVRENA and AFBVRENB, or AFBVRENC does require additional virtual storage in the LPA.

Each library module not in the applicable composite module is loaded from the VSF2LOAD library when the module is first referenced at run-time. However, these individual modules could be placed in the link pack area under their own module names, and then loaded when needed.

AFBVRENP cannot be customized. It can be placed in the LPA to reduce the region size required to run each parallel program, or it can be loaded into your region to avoid using extra storage in the LPA.

B.4 Tables of Required and Optional Modules

The approximate sizes and function descriptions for the required and optional composite modules provided by VS FORTRAN Version 2 are listed in the following figures:

Module	**Figure**
Required modules for AFBVRENA	Figure 36 in topic B.4.1
Optional modules for AFBVRENA	Figure 37 in topic B.4.1
Required modules for AFBVRENB	Figure 38 in topic B.4.2
Optional modules for AFBVRENB	Figure 39 in topic B.4.2
Required modules for AFBVRENC	Figure 40 in topic B.4.3
Optional modules for AFBVRENC	Figure 41 in topic B.4.3

The approximate module size for composite module AFBVRENP is listed on page B.4.4.

Subtopics

B.4.1 Composite Module AFBVRENA (MVS/XA or MVS/ESA)

Figure 36. Required Modules for AFBVRENA

Module	Approx. Size	Default Set	Function
AFBUOPT	D30	X	Error option table
AFBVABEX	1C68	X	ABEND processor
AFBVAREN	268	X	Internal linkage module
AFBVBLN$	80	X	Internal linkage routine
AFBVCDM$	80	X	Internal linkage module
AFBVCNI$	80	X	Internal linkage routine
AFBVCNO$	80	X	Internal linkage routine
AFBVCOM$	80	X	Internal linkage routine
AFBVCVT$	3E0	X	Internal linkage routine
AFBVDEB$	80	X	Internal linkage routine
AFBVDIO$	88	X	Internal linkage routine
AFBVDYN$	80	X	Internal linkage routine
AFBVEMG$	B8	X	Internal linkage routine
AFBVERE$	80	X	Internal linkage routine
AFBVERS$	A0	X	Internal linkage routine
AFBVFNTH	9D0	X	Program interrupt handler
AFBVGMFM	270	X	GETMAIN/FREEMAIN
AFBVGPRM	F0	X	Default run-time options
AFBVIAD$	80	X	Internal linkage routine
AFBVIIO$	88	X	Internal linkage routine
AFBVINI$	80	X	Internal linkage routine
AFBVINTP	2008	X	Initialization/termination
AFBVKIO$	88	X	Internal linkage routine
AFBVLBC0	2138	X	Library common work area
AFBVLOAD	480	X	Loader
AFBVLOC$	80	X	Internal linkage routine

AFBVMIN$	80	X	Internal linkage routine
AFBVMMA$	80	X	Internal linkage routine
AFBVMPR$	80	X	Internal linkage routine
AFBVPARM	17E8	X	Run-time options processor
AFBVPIO$	88	X	Internal linkage routine
AFBVPOS$	80	X	Internal linkage routine
AFBVRDCB	2A0	X	DCB information resolution
AFBVSPIE	1A0	X	Interrupt interceptor
AFBVSTAE	1C0	X	Abend control routine
AFBVSTIO	4D8	X	Standard I/O processor
AFBVTRC$	80	X	Internal linkage routine
AFBVTRMF	140	X	Termination of I/O
AFBVUAT(1)	D90	X	Unit attribute table
AFBVVIO$	C0	X	Internal linkage routine
Total	**C098**	**40**	

Note:

(1) May increase in size when the unit attribute table is customized.

Figure 37. Optional Modules for AFBVRENA

Module	**Approx. Size**	**Default Set**	**Function**
AFBDIOCP	1E0	X	Define file (LANGLVL 66)
AFBDSPAP(1)	560	X	Dimension calculator
AFBIBCOP(2)	D20	X	Pre-VS FORTRAN interface
AFBLDFIP(1)	700	X	List-directed I/O
AFBNAMEP	588	X	Namelist I/O
AFBPNPRP	D8		NPROCS processor
AFBSDUMQ	3940		SDUMP subroutine
AFBTFORP	158		Debugger interface
AFBVAMTP	1D8		Alt. error option table

			processor
AFBVASGP(3)	2040		DBCS assignment processor
AFBVASYP	1118	X	Asynchronous I/O
AFBVBALG	5C8	X	Boundary alignment routine (vector)
AFBVBLNT	3A8	X	Implied DO in I/O
AFBVCDMA	278	X	Common block directing maintenance
AFBVCLOP	478	X	CLOSE statement
AFBVCOMH	2030	X	Formatted I/O
AFBVCONI	330	X	Input floating-point conversion
AFBVCONO	840	X	Output floating-point conversion
AFBVCPTP	270	X	CPU time processing routine
AFBVCVTH	29F8	X	Data conversion
AFBVDBUP	1468		Debugging packet
AFBVDEBU	208		DEBUNIT parameter processor
AFBVDIVP	2330	X	Data in virtual processor
AFBVDOCP	168	X	Divide check and overflow test
AFBVDUMQ	8D8	X	DUMP/PDUMP subroutine
AFBVDYNA	9B0	X	Dynamic file allocation routine
AFBVEMGN	1510	X	Error message generator
AFBVERRE	268	X	Error summary
AFBVEXIP	128	X	Return code processor
AFBVFINP	C38	X	File information processor
AFBVFISC	370	X	File name scan
AFBVFMTP	260		Language conversion program define file
AFBVIIOS	370	X	Internal file services
AFBVINQP	20E8	X	INQUIRE statement
AFBVINTH	500	X	Vector program interrupt

			handler
AFBVIOCP	4E0	X	BACKSPACE, REWIND, ENDFILE
AFBVIOFP	EA0	X	Formatted I/O
AFBVIOLP	1EE0	X	List-directed I/O
AFBVIONP	1E58	X	Namelist I/O
AFBVIOUP	1520	X	Unformatted I/O
AFBVLINP	298	X	Link to reentrant CSECT
AFBVLOCA	8F0	X	Statement number locator
AFBVMOPP	660	X	Extended error handling
AFBVMPRM	368		AUTOTASK parameter processor
AFBVMSKL	8378	X	Message skeletons
AFBVOCMP	1C00	X	Obtain common blocks
AFBVOPEP	1CA8	X	OPEN statement
AFBVPOSA	4168		Post ABEND processor
AFBVSCOP(4)	9D0	X	Pre-Release 4 interface
AFBVSPAP	8A8	X	Array dimension calculator
AFBVSPIP	4B0	X	Dynamic spill area processor (vector)
AFBVTEN	2C0	X	Powers of ten table
AFBVTIMP	6E8	X	Date/time/clock routine
AFBVTRCH	EA8	X	Traceback generator
AFBVUNIN	220	X	Unnormalized operand interrupt handler
AFBVVINI	390	X	Vector initialization

Note:

(1) These modules are used for the specified functions that are performed from object decks produced by Fortran compilers prior to VS FORTRAN Version 1 Release 4.

(2) Module AFBIBCOP is used when running object decks produced by Fortran compilers prior to VS FORTRAN. It is needed for formatted and unformatted I/O and for initialization from a main program.

(3) Included in AFBVASGP are AFBDBMOV, AFBDBGVB, AFBDBMVE, AFBDBPAD, AFBDBTRC, AFBDBTRT. The size of AFBVASGP also incorporates the sizes of the other modules.

(4) The module AFBVSCOP is used when running object decks produced by the VS FORTRAN Version 1 compiler from releases prior to Release 4. It is needed for formatted and unformatted I/O and for initialization from a main program or from a subroutine with character arguments.

B.4.2 Composite Module AFBVRENB (MVS/XA or MVS/ESA)

Figure 38. Required Modules for AFBVRENB

Module	Approx. Size	Default Set	Function
AFBVBREN	E0	X	Internal linkage module
AFBVFIST	1628	X	File status processor
AFBVSIOS	53A8	X	Sequential I/O services
Total	**6AB0**	**3**	

Figure 39. Optional Modules for AFBVRENB

Module	Approx. Size	Default Set	Function
AFBVASUB	E58		Asynchronous I/O
AFBVDIOS	24F0		Direct access I/O services
AFBVKIOS	3278		Keyed access I/O services
AFBVMSTP	190		MTF subtask control
AFBVPIOS	2EF0		Sequential striped I/O processing
AFBVVIOS	2170		Nonkeyed VSAM I/O services

B.4.3 Composite Module AFBVRENC (MVS/SP Version 1)

Figure 40. Required Modules for AFBVRENC

Module	Approx. Size	Default Set	Function
AFBUOPT	D30	X	Error option table
AFBVABEX	1C68	X	ABEND processor
AFBVBLN$	80	X	Internal linkage routine
AFBVCDM$	80	X	Internal linkage module
AFBVCNI$	80	X	Internal linkage routine
AFBVCNO$	80	X	Internal linkage routine
AFBVCOM$	80	X	Internal linkage routine
AFBVCVT$	3E0	X	Internal linkage routine
AFBVDEB$	80	X	Internal linkage routine
AFBVDIO$	88	X	Internal linkage routine
AFBVDYN$	80	X	Internal linkage routine
AFBVEMG$	B8	X	Internal linkage routine
AFBVERE$	80	X	Internal linkage routine
AFBVERS$	A0	X	Internal linkage routine
AFBVFIST	1628	X	File status processor
AFBVFNTH	9D0	X	Program interrupt handler
AFBVGMFM	270	X	GETMAIN/FREEMAIN
AFBVGPRM	F0	X	Default run-time options
AFBVIAD$	80	X	Internal linkage routine
AFBVIIO$	88	X	Internal linkage routine
AFBVINI$	80	X	Internal linkage routine
AFBVINTP	2008	X	Initialization/termination
AFBVKIO$	88	X	Internal linkage routine
AFBVLBC0	2138	X	Library common work area
AFBVLOAD	480	X	Loader
AFBVLOC$	80	X	Internal linkage routine

AFBVMIN$	80	X	Internal linkage routine
AFBVMMA$	80	X	Internal linkage routine
AFBVMPR$	80	X	Internal linkage routine
AFBVPARM	17E8	X	Run-time options processor
AFBVPIO$	88	X	Internal linkage routine
AFBVPOS$	80	X	Internal linkage routine
AFBVRDCB	2A0	X	DCB information processor
AFBVREN	2A8	X	Internal linkage module
AFBVSIOS	53A8	X	Sequential I/O services
AFBVSPIE	1A0	X	Interrupt interceptor
AFBVSTAE	1C0	X	Abend control routine
AFBVSTIO	4D8	X	Standard I/O processor
AFBVTRC$	80	X	Internal linkage routine
AFBVTRMF	140	X	Termination of I/O
AFBVUAT(1)	D90	X	Unit attribute table
AFBVVIO$	C0	X	Internal linkage routine
Total	**12AA8**	**42**	

Note:

(1) May increase in size when the unit attribute table is customized.

Figure 41. Optional Modules for AFBVRENC

Module	**Approx. Size**	**Default Set**	**Function**
AFBDIOCP	1E0		Define file (LANGLVL 66)
AFBDSPAP(1)	560		Dimension calculator
AFBIBCOP(2)	D20		Pre-VS FORTRAN interface
AFBLDFIP(1)	700		List-directed I/O
AFBNAMEP(1)	588		Namelist I/O
AFBPNPRP	D8		NPROCS processor
AFBSDUMQ	3940		SDUMP subroutine

AFBTFORP	158		Debugger interface
AFBVAMTP	1D8		Alt. error option table processor
AFBVASGP(3)	2040		DBCS assignment processor
AFBVASUB	E58		Asynchronous I/O
AFBVASYP	1118		Asynchronous I/O
AFBVBALG	5C8		Boundary alignment routine (Vector)
AFBVBLNT	3A8	X	Implied DO in I/O
AFBVCDMA	278		Common block directory maintenance
AFBVCLOP	478	X	CLOSE statement
AFBVCOMH	2030	X	Formatted I/O
AFBVCONI	330	X	Input floating-point conversion
AFBVCONO	840	X	Output floating-point conversion
AFBVCPTP	270		CPU time processing routine
AFBVCVTH	29F8	X	Data conversion
AFBVDBUP	1468		Debugging packet
AFBVDEBU	208		DEBUNIT parameter processor
AFBVDIOS	24F0		Direct access I/O services
AFBVDOCP	168		Divide check and overflow test
AFBVDUMQ	8D8		DUMP/PDUMP subroutine
AFBVDYNA	9B0		DFA processor
AFBVEMGN	1510	X	Error message generator
AFBVERRE	268	X	Error summary
AFBVEXIP	128		Return code processor
AFBVFINP	C38		File information processor
AFBVFISC	370		File name scan
AFBVFMTP	260		Language conversion program define file
AFBVIIOS	370		Internal file services
AFBVINQP	20E8		INQUIRE statement
AFBVINTH	500		Vector program interrupt handler

AFBVIOCP	4E0		BACKSPACE, REWIND, ENDFILE
AFBVIOFP	EA0	X	Formatted I/O
AFBVIOLP	1EE0	X	List-directed I/O
AFBVIONP	1E58		Namelist I/O
AFBVIOUP	1520	X	Unformatted I/O
AFBVKIOS	3278		Keyed access I/O services
AFBVLINP	298		Link to reentrant CSECT
AFBVLOCA	8F0	X	Statement number locator
AFBVMOPP	660		Extended error handling
AFBVMPRM	368		AUTOTASK parameter processor
AFBVMSKL	8378	X	Message skeletons
AFBVOCMP	1C00		Obtain common blocks
AFBVOPEP	1CA8	X	OPEN statement
AFBVPIOS	2EF0		Sequential striped I/O processing
AFBVPOSA	4168		Post ABEND processor
AFBVSCOP(4)	9D0		Pre-Release 4 interface
AFBVSPAP	8A8		Array dimension calculator
AFBVSPIP	4B0		Dynamic spill area processor (vector)
AFBVTEN	2C0	X	Powers of ten table
AFBVTIMP	6E8		Date/time/clock routine
AFBVTRCH	EA8	X	Traceback generator
AFBVUNIN	220		Unnormalized operand interrupt handler
AFBVVINI	390		Vector initialization
AFBVVIOS	2170		Nonkeyed VSAM I/O services

Note:

(1) These modules are used for the specified functions that are performed from object decks produced by Fortran compilers prior to VS FORTRAN Version 1 Release 4.

(2) Module AFBIBCOP is used when running object decks produced by Fortran compilers prior to VS FORTRAN. It is needed for formatted and unformatted I/O and for initialization from a main program.

(3) Included in AFBVASGP are AFBDBMOV, AFBDBGVB, AFBDBMVE, AFBDBPAD, AFBDBTRC, AFBDBTRT. The size of AFBVASGP also incorporates the sizes of the other modules.

(4) The module AFBVSCOP is used when running object decks produced by the VS FORTRAN Version 1 compiler from releases prior to Release 4. It is needed for formatted and unformatted I/O and for initialization from a main program or from a subroutine with character arguments.

B.4.4 Composite Module AFBVRENP (Any MVS)

The composite module AFBVRENP cannot be customized. The approximate size for the module is X'12B78'.

C.0 Appendix C. Servicing VS FORTRAN Version 2

IBM National Services Division (NSD) provides corrective and preventive service for product defects, as well as support for resolving program problems, through Central Service, including the IBM Support Center. For details of these facilities and a list of all the products supported, refer to *Field Engineering Programming System General Information Manual*.

Subtopics

C.1 Problem Reporting

When you encounter a failure in the product, follow this procedure:

1. Use the *VS FORTRAN Version 2 Diagnosis Guide* to assist you in describing the failure as a keyword string.

2. Compare that keyword string with the index of keywords in the software support data base of documented failures. Contact the IBM Support Center for assistance in the keyword search if necessary.

3. If you cannot find a documented failure similar to yours, report your failure to the IBM Support Center as an authorized programming analysis report (APAR).

   An APAR is resolved by Central Service with either an explanation or a new corrective service program temporary fix (PTF) for the defect. A PTF is a replacement text module that is installed in the product to correct the defect. Collections of new PTFs for products are provided to all customers as preventive service program update tapes (PUTs).

¦ With VS FORTRAN Version 2 Release 6, PTFs will also include updates to the README file, which is updated with new information ¦ pertaining to this product as the result of service updates, as well as any additional information provided in response to Reader ¦ Comment Forms.

When reporting a problem, you may need the Function Modifier Identifier (FMID) and Field Engineering Service Number (FESN). These numbers are shown in Figure 42 and in Figure 43.

Figure 42. Function Modifier Identifiers (FMIDs)

FMID	Definition
HFR2602	Interactive debug (IAD) modules common to MVS and VM
JFR2620	IAD panels for ISPF support under MVS
JFR2611	IAD TSO Help
HFT2602	Compiler modules common to MVS and VM
JFT2611	Compiler modules unique to MVS
HFL2602	Library modules common to MVS and VM
JFL2611	Library modules unique to MVS
JFT2612	PostScript Publication files
JFT2613	BookMaster Publication files

Figure 43. Field Engineering Service Numbers

FMID	Component ID	FESN	System
HFR2602	5668-80602	6580602	MVS
JFR2620	5668-80602	6580602	MVS
JFR2611	5668-80602	6580602	MVS

HFT2602	5668-80601	6580601	MVS
JFT2611	5668-80601	6580601	MVS
JFT2612	5668-80601	N/A	MVS
JFT2613	5668-80601	N/A	MVS
HFL2602	5668-80501	6580501	MVS
JFL2611	5668-80501	6580501	MVS

C.2 Corrective Service

Corrective service is distributed on a PTF to a specific customer, and contains a correction for a single known problem in a particular product. As soon as a new problem is identified and the solution is established, a PTF is created and made available on a corrective service tape.

Enclosed with each corrective service tape is a Memo to Users, which contains a list of tape contents, and a cover letter for each PTF on the tape. Service is distributed in a format acceptable to SMP/E.

#Following is an example of the series of steps required to complete the installation of a PTF. You should always refer to the #documentation accompanying the tape for accurate instructions for installation. These jobs are included as examples of the jobs #which are provided to facilitate installation. In these jobs, the file designated as **File 1** contains the actual PTF, and **File 3** contains #related prerequisites, corequisites, and/or if-requisites. Note that **File 3** is not necessarily included in every PTF.

To RECEIVE, APPLY and ACCEPT service, use SMP/E as follows:

1. RECEIVE the service using the sample job in Figure 44.

```
-------------------------------------------------------------------------

  //RECEIVE      JOB
  //SMP          EXEC AFFEPROC                 Note 1
  //SMPPTFIN DD  DISP=OLD,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200),
  //             DSN=PTF,VOL=SER=PTFTPE,LABEL=(1,NL),UNIT=3400-6
#  //         DD  DISP=OLD,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200),
  //             DSN=PTF,VOL=SER=PTFTPE,LABEL=(3,NL),UNIT=3400-6
  //SMPCNTL  DD  *
       SET BDY (GLOBAL) .
       RECEIVE SOURCEID(source-id)           Note 2
               SYSMODS .
  /*
```

Notes:

1. To install the compiler and library product, use ILXEPROC instead of AFFEPROC. For the library-only, use AFBEPROC instead of AFFEPROC.

2. source-id is a 1 to 8 character string that SMP/E will associate with all PTFs processed in this step.

```
-------------------------------------------------------------------------
Figure 44. Job for SMP/E to Receive Corrective Service
```

2. APPLY the service using the sample job in Figure 45.

```
-------------------------------------------------------------------------

  //APPLY       JOB
  //SMP          EXEC AFFEPROC                Note 1
  //SMPCNTL  DD  *
       SET BDY (TVSF2) .
       APPLY SOURCEID(source-id).             Note 2
  /*
```

Notes:

1. To install corrective service for the compiler and library product,

use ILXEPROC instead of AFFEPROC. For the library-only, use AFBEPROC instead of AFFEPROC.

2. source-id is a 1 to 8 character string that SMP/E will associate with all PTFs processed in this step.

Figure 45. Job for SMP/E to Apply Corrective Service

3. ACCEPT the service using the sample job in Figure 46.

```
//ACCEPT       JOB
//SMP          EXEC AFFEPROC                Note 1
//SMPCNTL  DD  *
    SET BDY (DVSF2) .
    ACCEPT SOURCEID(source-id).             Note 2
/*
```

Notes:

1. To ACCEPT corrective service for the compiler and library product, use ILXEPROC instead of AFFEPROC. For the library-only, use AFBEPROC instead of AFFEPROC.

2. source-id is a 1 to 8 character string that SMP/E will associate with all PTFs processed in this step.

Figure 46. Job for SMP/E to Accept Corrective Service

C.3 Preventive Service

Preventive service is distributed on PUTs to all customers on a regular basis. They are made up of PTFs to problems in all IBM products that operate under your operating system. Note that an MVS PUT contains only the latest PTFs. Therefore, to bring your Fortran system up to the latest level, you must run all prior PUT tapes for the product before you run the most recent tape.

To ACCEPT preventive service, follow the same steps used to ACCEPT corrective service using SMP/E.

Numerics

A

BACK_1 Communicating Your Comments to IBM
VS FORTRAN Version 2 Installation and Customization for MVS Release 6

Publication No. SC26-4340-05

If there is something you like--or dislike--about this book, please let us know. You can use one of the methods listed below to send your comments to IBM. If you want a reply, include your name, address, and telephone number. If you are communicating electronically, include the book title, publication number, page number, or topic you are commenting on.

The comments you send should only pertain to the information in this book and its presentation. To request additional publications or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give it to the local IBM branch office or IBM representative for postage-paid mailing.

If you prefer to send comments by mail, use the RCF at the back of this book

If you prefer to send comments by FAX, use this number

- United States and Canada: 416-448-6161

- Other countries: (+1)-416-448-6161

If you prefer to send comments electronically, use the network ID listed below. Be sure to include your entire network address if you wish a reply.

- Internet: torrcf@vnet.ibm.com
- IBMLink: toribm(torrcf)
- IBM/PROFS: torolab4(torrcf)
- IBMMAIL: ibmmail(caibmwt9)

COMMENTS Readers' Comments -- We'd Like to Hear from You
VS FORTRAN Version 2 Installation and Customization for MVS Release 6

Publication No. SC26-4340-05

Overall, how satisfied are you with the information in this book?

Legend:

1 Very satisfied
2 Satisfied
3 Neutral
4 Dissatisfied
5 Very dissatisfied

	1	2	3	4	5
Overall satisfaction					

How satisfied are you that the information in this book is:

	1	2	3	4	5
Accurate					
Complete					
Easy to find					
Easy to understand					
Well organized					
Applicable to your tasks					

Please tell us how we can improve this book:

IBM Canada Ltd. Laboratory
Information Development
2G/345/1150/TOR
1150 EGLINTON AVENUE EAST
NORTH YORK ONTARIO CANADA M3C 1H7

Name . . . . . _____
Company or Organization _____
Address . . . . . _____
_____
_____
Phone No. . . . . _____